Fast and backward stable computation of roots of polynomials, Part IIa: general backward error analysis

Jared L. Aurentz
Thomas Mach
Leonardo Robol
Raf Vandebril
David S. Watkins
Report TW683, October 2017

KU Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

Fast and backward stable computation of roots of polynomials, Part IIa: general backward error analysis

Jared L. Aurentz
Thomas Mach
Leonardo Robol
Raf Vandebril
David S. Watkins
Report TW 683, October 2017

Department of Computer Science, KU Leuven

Abstract

This report is a continuation of *Fast and backward stable computation of roots of polynomials* by J. L. Aurentz, T. Mach, R. Vandebril, and D.S. Watkins, SIAM Journal on Matrix Analysis and Applications, 36(3): 942–973, 2015. It is also an early version of the paper *Fast and backward stable computation of roots of polynomials, Part II: backward error analysis; companion matrix and companion pencil*, which we have submitted for publication. We introduce a companion QZ algorithm that computes the roots of a polynomial by solving a generalized eigenvalue problem for a companion pencil. More importantly, we provide an improved backward error analysis that exploits the special structure of the companion matrix or pencil. This report is nearly identical to the paper that we eventually submitted, except for the backward error analysis in Section 5.

FAST AND BACKWARD STABLE COMPUTATION OF ROOTS OF POLYNOMIALS, PART IIA: GENERAL BACKWARD ERROR ANALYSIS*

JARED L. AURENTZ†, THOMAS MACH¶, LEONARDO ROBOL‖, RAF VANDEBRIL‡, AND DAVID S. WATKINS§

Abstract. This report is a continuation of *Fast and backward stable computation of roots of polynomials* by J. L. Aurentz, T. Mach, R. Vandebril, and D.S. Watkins, SIAM Journal on Matrix Analysis and Applications, 36(3): 942–973, 2015. It is also an early version of the paper *Fast and backward stable computation of roots of polynomials, Part II: backward error analysis; companion matrix and companion pencil*, which we have submitted for publication. We introduce a companion QZ algorithm that computes the roots of a polynomial by solving a generalized eigenvalue problem for a companion pencil. More importantly, we provide an improved backward error analysis that exploits the special structure of the companion matrix or pencil. This report is nearly identical to the paper that we eventually submitted, except for the backward error analysis in Section 5.

Key words. polynomial, root, companion matrix, companion pencil, eigenvalue, Francis algorithm, QR algorithm, QZ algorithm, core transformation, backward stability

AMS subject classifications. 65F15, 65H17, 15A18, 65H04

1. Introduction.

The reason for this report. This technical report is a version of [3] that we were going to publish until we realized that by changing one line in one key subroutine we suddenly had a better algorithm that admits a stronger and simpler backward error analysis. The subroutine in question executes the *turnover*, a key operation in our algorithm that is described in the paper. A certain scalar that we call ρ (defined in the paper) is invariant throughout the course of our algorithm, assuming exact arithmetic. In practice, i.e. in floating-point arithmetic ρ acquires a backward error $\delta\rho$ that must be taken into account in the backward error analysis. Any reasonably designed turnover is normwise backward stable and will produce a backward error $\delta\rho$ that is tiny in an absolute sense, but might not be tiny relative to ρ if ρ itself is very small. This report provides the backward error analysis under this assumption.

Just as we were preparing this material for publication we realized that by changing one line in the turnover code we obtain an improved turnover that yields a backward error $\delta\rho$ that is tiny not just in an absolute sense but also relative to ρ. This results in a more accurate algorithm that also admits a simpler and stronger backward error analysis. This is the subject of [3].

We felt it was worth preserving the older (and more complicated) error analysis because it is innovative and interesting in its own right, and the techniques developed

* The research was partially supported by the Research Council KU Leuven, projects C14/16/056 (Inverse-free Rational Krylov Methods: Theory and Applications).

†Instituto de Ciencias Matemáticas, Universidad Autónoma de Madrid, Madrid, Spain (JaredAurentz@gmail.com).

‡Department of Computer Science, University of Leuven, KU Leuven, Leuven, Belgium; (Raf.Vandebril@cs.kuleuven.be).

‖Istituto di Scienza e Tecnologie dell Informazione 'A. Faedo' (ISTI), CNR, Pisa, Italy; (Leonardo.Robol@isti.cnr.it).

§Department of Mathematics, Washington State University, Pullman, WA 99164-3113, USA; (watkins@math.wsu.edu).

¶Department of Mathematics, School of Science and Technology, Nazarbayev University, 010000 Astana, Kazakhstan; (thomas.mach@nu.edu.kz).

there might be useful elsewhere. Also, it is nice to know that a satisfactory error analysis is possible even when using a turnover that does not guarantee that $\delta\rho$ is small relative to ρ.

Introduction. Consider the problem of computing the n zeros of a complex polynomial

$$p(z) = a_n z^n + a_{n-1} z^{n-1} + \cdots + a_1 z + a_0, \quad a_n \neq 0, \quad a_0 \neq 0,$$

expressed in terms of the monomial basis. One way to do this is to form the companion matrix

$$A = \begin{bmatrix} & & -a_0/a_n \\ 1 & & -a_1/a_n \\ & \ddots & \vdots \\ & 1 & -a_{n-1}/a_n \end{bmatrix}, \tag{1.1}$$

and compute its eigenvalues. This is what MATLAB's `roots` command does. But `roots` does not exploit the special structure of the companion matrix, so it requires $O(n^2)$ storage (one matrix) and $O(n^3)$ flops using Francis's implicitly-shifted QR algorithm [15]. It is natural to ask whether we can save space and flops by exploiting the structure. This has, in fact, been done [7, 10, 11], and we have also made a contribution [4]. All of the methods proposed in these papers use the unitary-plus-rank-one structure of the companion matrix to build a special data structure that brings the storage down to $O(n)$. Then Francis's algorithm, operating on the special data structure, has a flop count of $O(n^2)$. Based on our tests [4], our method appears to be the fastest of the several that have been proposed. Moreover, we are the only ones who have been able to prove that our method is backward stable. In this paper we will refer to our method as the *companion QR algorithm*.

In cases where the polynomial has a particularly small leading coefficient, one might hesitate to use the companion matrix, since division by a tiny a_n will result in very large entries in (1.1). This could aversely affect accuracy, one might think. An alternative to division by a_n is to work with a *companion pencil*

$$A - \lambda B = \begin{bmatrix} 0 & & & -a_0 \\ 1 & 0 & & -a_1 \\ & \ddots & \ddots & \vdots \\ & & 1 & -a_{n-1} \end{bmatrix} - \lambda \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & a_n \end{bmatrix}, \tag{1.2}$$

which also has the roots of p as its eigenvalues. More generally we can consider any pencil of the form

$$V - \lambda W = \begin{bmatrix} 0 & & & -v_1 \\ 1 & & & -v_2 \\ & 1 & & -v_3 \\ & & \ddots & \vdots \\ & & 1 & -v_n \end{bmatrix} - \lambda \begin{bmatrix} 1 & & & w_1 \\ & 1 & & w_2 \\ & & \ddots & \vdots \\ & & 1 & w_{n-1} \\ & & & w_n \end{bmatrix}, \tag{1.3}$$

where

$$\begin{aligned} v_1 &= a_0, \\ v_{i+1} + w_i &= a_i, \text{ for } i = 1, \ldots, n-1, \text{and} \\ w_n &= a_n. \end{aligned} \tag{1.4}$$

One easily checks that all pencils of this form also have the zeros of p as their eigenvalues [5, 18]. This generalized eigenvalue problem is in Hessenberg-triangular form and can be solved by the Moler-Stewart variant [20, 26, 27] of Francis's algorithm, commonly called the QZ algorithm. In this paper we introduce a generalization of the method of [4] to matrix pencils, which we will call the *companion QZ algorithm*. This is a straightforward exercise, and it is not the main point of this publication.

This paper is really about backward error analysis. After the publication of [4] we realized that the analysis in that paper was not quite right. There is a factor that is constant in exact arithmetic and that we treated as a constant. We should have taken into account the backward error in that factor, as this makes a difference in the analysis. When we wrote an earlier version of this paper (with different title and emphasis), we took the opportunity to repair that error. We were also able to improve the analysis by taking into account the structure of the backward error that follows from the structure of the companion matrix or pencil. (Un)fortunately that paper was rejected. We were annoyed at first, but now we are happy for the opportunity to write a better paper.

When we introduced (in the rejected paper) the companion pencil generalization, we had no doubt at all that there would be cases where the companion QR method fails but companion QZ succeeds: just take a polynomial with a tiny leading coefficient! We were so sure of this that we failed to include an example in the paper. What an oversight! As we began to make revisions in preparation for resubmission, we looked for examples where companion QZ succeeds and companion QR fails. Here we got a surprise: we couldn't find any such examples. The companion QR method is much more robust than we had realized.

This discovery led us to take a closer look at the backward error analysis and try to explain why the companion QR algorithm works so well. This paper is the result of that investigation. We prove that, assuming $\|A\|$ is less than the reciprocal of the unit roundoff (typically 10^{16}), the companion matrix method is just as good as the companion pencil method. Both methods are significantly more accurate than a method like MATLAB's `roots` that computes the eigenvalues of the companion matrix without exploiting the special structure. This is the main message of this paper.

The paper is organized as follows. Section 2 briefly discusses previous work in this area. The memory-efficient $O(n)$ factorization of a companion pencil $V - \lambda W$ of the form (1.3) is introduced in Section 3, together with some necessary background information. In Section 4 we introduce the companion QZ algorithm and demonstrate its $O(n^2)$ performance.

The heart of the paper is the backward error analysis in Section 5. The main results are as follows: Let p be a polynomial with coefficient vector $a = \begin{bmatrix} a_0 & \cdots & a_n \end{bmatrix}$. Suppose we compute the zeros of p by some method, and let $\hat{p}$, with coefficient vector $\hat{a}$, be an *appropriately scaled* polynomial that has the computed roots as its exact zeros. The *absolute normwise backward error on the coefficients* is $\|a - \hat{a}\|$. If our companion QR method is used to do the computation, the backward error satisfies $\|a - \hat{a}\| \lesssim u\|a\|$, where u is the unit roundoff. This is an optimal result, and it is

better than can be achieved by the unstructured Francis algorithm applied to the companion matrix. The latter gives only $\|a - \hat{a}\| \lesssim u\|a\|^2$ (see Figure 5.6). If the companion *pencil* is used, we do not get the optimal result unless we apply the companion QZ algorithm (or any stable algorithm) to a rescaled polynomial $p/\|a\|$. If we do this, we get the optimal result $\|a - \hat{a}\| \lesssim u\|a\|$.

2. Earlier work. There are many ways [19] to compute roots of polynomials. Here we focus on companion matrix and pencil methods. Computing roots of polynomials in the monomial basis via the companion matrix has been the subject of study of several research teams. See [4] for a summary.

There is, to our knowledge, only one article by Boito, Eidelman, and Gemignani [8] presenting a structured QZ algorithm for computing roots of polynomials in the monomial basis. The authors consider a matrix pencil, say (V, W), where both V and W are of unitary-plus-rank-one form, V is Hessenberg and W is upper triangular. Both matrices are represented efficiently by a quasiseparable representation. To counter the effects of roundoff errors, some redundant quasiseparable generators to represent the unitary part are created; a compression algorithm to reduce the number of generators to a minimal set is presented. The computational cost of each structured QZ iteration is $O(n)$. A double shift version of this algorithm is presented by Boito, Eidelman, and Gemignani in [9].

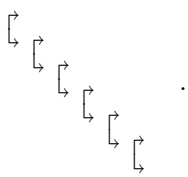
The companion pencil (1.2) is the most frequently appearing one in the literature, but, there is a wide variety of comparable matrix pencils with the same eigenvalues [6, 13, 18], many of which are highly structured. In this article we will focus on companion pencils of the form (1.3).

3. Core transformations and factoring companion matrices. Core transformations will be used throughout the paper and are the building blocks for a fast algorithm and an efficient representation of the companion pencil.

3.1. Core transformations. A nonsingular matrix G_i identical to the identity matrix except for a 2×2 submatrix in position $(i : i+1, i : i+1)$ is called a *core transformation*. The subscript i refers to the position of the diagonal block $(i : i+1, i : i+1)$ called the *active part* of the core transformation. Core transformations G_i and G_j commute if $|i - j| > 1$.

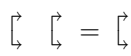
In previous work [1, 2] the authors have used non-unitary core transformations, but here we will only use unitary core transformations. Thus, in this paper, the term *core transformation* will mean *unitary* core transformation; the active part could be a rotator or a reflector, for example.

To avoid excessive index usage, and to ease the understanding of the interaction of core transformations, we depict them as $\vec{\hookrightarrow}$, where the tiny arrows pinpoint the active part. For example, every unitary upper Hessenberg matrix Q can be factored as the product of $n-1$ core transformations in a *descending* order $Q = G_1 G_2 \cdots G_{n-1}$. Such a *descending sequence* of core transformations is represented pictorially by



All of the algorithms in this paper are described in terms of core transformations and two operations: the *fusion* and the *turnover*.

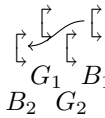
Fusion. The product of two unitary core transformations G_i and H_i is again a unitary core transformation. Pictorially we can write this as



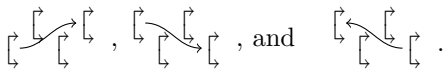
Turnover. The product of three core transformations $F_iG_{i+1}H_i$ is a 3×3 unitary matrix that can be factored also as $F_{i+1}G_iH_{i+1}$, depicted as

$$\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} =$$

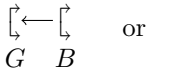
Pictorial action of a turnover and fusion. We will see, when describing the algorithms, that there are typically one or more core transformations not fitting the pattern (called the *misfit(s)*), that need to be moved around by executing turnovers and similarities. To describe clearly the movement of the misfit we use the following pictorial description:



where B_1 is the misfit before the turnover and B_2 after the turnover. The core transformations G_1 and G_2 are involved in the turnover and change once B_2 is created. The picture is mathematically equivalent to $G_1G_2B_1 = B_2\hat{G}_1\hat{G}_2$. Other possible turnovers are



Directly at the start and at the end of each QZ (and QR) iteration, a misfit is fused with another core transformation, so that it vanishes. We will describe this pictorially as



where B is to be fused with G.

3.2. A factorization of the companion pencil. The pencil matrices V and W from (1.3) are both unitary-plus-rank one, V is upper Hessenberg and W is upper triangular. We store V in QR decomposed form: $V = QR$, where Q is unitary and upper Hessenberg, and R is upper triangular and unitary-plus-rank-one. In fact

$$Q = \begin{bmatrix} 0 & & & & 1 \\ 1 & & & & 0 \\ & 1 & & & 0 \\ & & \ddots & & \vdots \\ & & & 1 & 0 \end{bmatrix} \quad \text{and} \quad R = \begin{bmatrix} 1 & & & & -v_2 \\ & 1 & & & -v_3 \\ & & \ddots & & \vdots \\ & & & 1 & -v_n \\ & & & & -v_1 \end{bmatrix}. \tag{3.1}$$

We need efficient representations of Q, R, and W. Q is easy; it is the product of $n-1$ core transformations: $Q = Q_1 \cdots Q_{n-1}$, with $Q_i(i:i+1, i:i+1) = \left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]$.

Factoring an upper triangular unitary-plus-rank-one matrix. The matrices R (3.1) and W (1.3) have exactly the same structure and can be factored in the same way. We focus on R. It turns out that for this factorization we need to add a bit of room by adjoining a row and column. Let

$$\underline{R} = \left[\begin{array}{cccc|c} 1 & & & -v_2 & 0 \\ & 1 & & -v_3 & 0 \\ & & \ddots & \vdots & \vdots \\ & & 1 & -v_n & 0 \\ & & & -v_1 & 1 \\ \hline & & & 0 & 0 \end{array}\right]. \tag{3.2}$$

This is just R with a zero row and nearly zero column added. The 1 in the last column ensures that $\underline{R}$ is unitary-plus-rank-one: $\underline{R} = Y_n + \underline{z} e_n^T$, where

$$Y_n = \left[\begin{array}{cccc|c} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \vdots \\ & & & 1 & \\ & & & 0 & 1 \\ \hline & & & 1 & 0 \end{array}\right] \quad \text{and} \quad \underline{z} = -\left[\begin{array}{c} v_2 \\ v_3 \\ \vdots \\ v_n \\ v_1 \\ \hline 1 \end{array}\right].$$

Let $C_1, \ldots, C_n$ be core transformations such that $C\underline{z} = C_1 \cdots C_n \underline{z} = \alpha e_1$, where $|\alpha| = \|\underline{z}\|_2$. Let B be the unitary Hessenberg matrix $B = CY_n$. Clearly $B = B_1 \cdots B_n$, where $B_i = C_i$ for $i = 1, \ldots, n-1$, and $B_n = C_n Y_n$. This gives us a factorization of $\underline{R}$ as

$$\underline{R} = C_n^* \cdots C_1^* (B_1 \cdots B_n + \alpha\, e_1 \underline{y}^T) = C^* (B + \alpha\, e_1 \underline{y}^T), \tag{3.3}$$

with $\underline{y}^T = e_n^T \in \mathbb{R}^{n+1}$. In the course of our algorithm, the core transformations B_i, C_i, and the vector $\underline{y}$ will be modified repeatedly, but the form (3.3) for $\underline{R}$ will be preserved. $\underline{R}$ remains upper triangular with its last row equal to zero. The theory that supports these claims can be found in [4, § 4]. Notice that the core transformations B_n and C_n both make use of row and column $n+1$. Had we not added a row and column, this factorization would not have been possible.

Multiplying (3.3) on the left by e_{n+1}^T, we find that $0 = e_{n+1}^T \underline{R} = e_{n+1}^T C^* B + \alpha\, e_{n+1}^T C^* e_1 \underline{y}^T$, so [4, Thm. 4.6]

$$\alpha\, \underline{y}^T = -(e_{n+1}^T C^* e_1)^{-1} e_{n+1}^T C^* B. \tag{3.4}$$

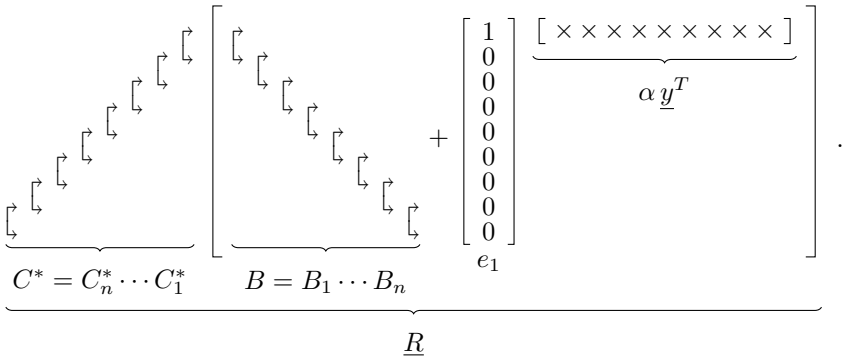
This equation demonstrates that the information about $\alpha\, \underline{y}^T$, which determines the rank-one part, is encoded in the core transformations. This means that we will be able to develop an algorithm that does not keep track of $\underline{y}$; the rank-one part can be simply ignored. If at any time we should need $\underline{y}$ or some part of $\underline{y}$, we could recover it from C and B using (3.4). However, as we show in [4, § 4], it turns out that we never need to use (3.4) in practice.

Let $P = I_{(n+1)\times n}$, the $(n+1) \times n$ matrix obtained by deleting the last column from the $(n+1) \times (n+1)$ identity matrix. Then $R = P^T \underline{R} P$, so our factored form of

R is

$$R = P^T C_n^* \cdots C_1^* (B_1 \cdots B_n + \alpha\, e_1 \underline{y}^T) P. \tag{3.5}$$

The matrices P and P^T are included just so that the dimensions of R come out right. They play no active role in the algorithm, and we will mostly ignore them. Pictorially, for $n = 8$, $\underline{R}$ (and hence also R) can be represented as



Since we can ignore $\alpha\, \underline{y}^T$, we see that R is represented by two sequences of core transformations, $\mathcal{C}$ and $\mathcal{B}$. Hence $A = QR$ is represented by three sequences of core transformations.

The matrix W admits a factorization of the same form as (3.5)

$$W = P^T C_W^* (B_W + \alpha_W e_1 \underline{y}_W^T) P,$$

with $C_W[w_1, \ldots, w_n, 1]^T = \alpha_W e_1$.

Thus W is also represented by two sequences of core transformations. Altogether the pencil (V, W) is represented by five sequences of core transformations.

3.3. Core transformations and upper triangular matrices. In the next section we will show how to compute the eigenvalues of the matrix pencil (V, W) via the QZ algorithm as described by Vandebril and Watkins [25]. An important operation is to refactor the product of an upper triangular matrix times a core transformation RG_i as the product of a core transformation times an upper triangular matrix[1] $\hat{G}_i \hat{R}$. The other way proceeds similarly. Considering dense matrices we get pictorially

Applying a nontrivial core transformation from the right on the upper triangular matrix creates a non-zero subdiagonal, which can be removed by pulling out a nontrivial core transformation from the left.

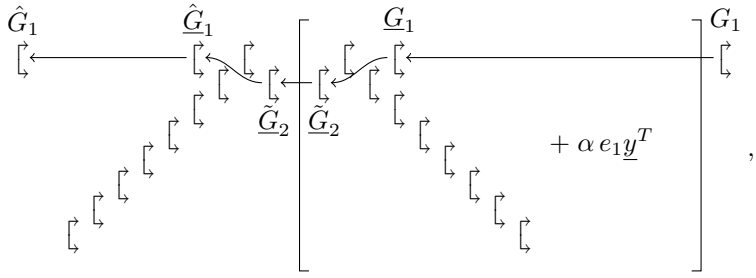
In our case the upper triangular matrix is represented in a data-sparse way (3.3). Instead of explicitly creating the upper triangular matrix to interchange the order of the upper triangular matrix and a core transformation we operate on the factored form directly. There are two versions, *passing a core transformation* from right to left and vice versa.

[1] We assume that eigenvalues at zero or infinity are deflated beforehand, so the involved upper triangular matrices are nonsingular.

From right to left. Consider an arbitrary upper triangular unitary-plus-rank-one matrix, say, $R = P^T C^*(B + \alpha\, e_1 \underline{y}^T)P$. Let G_i $(1 \leq i \leq n-1)$ be the core transformation we want to move from right to left. We have $PG_i = \underline{G}_i P$, where $\underline{G}_i$ is the $(n+1) \times (n+1)$ version of G_i. We have $RG_i = P^T C^*(B + \alpha\, e_1 \underline{y}^T)\underline{G}_i P = P^T C^*(B\underline{G}_i + \alpha\, e_1 \underline{y}^T \underline{G}_i)P$, so $\underline{y}^T$ changes into $\hat{\underline{y}}^T = \underline{y}^T \underline{G}_i$. The next step is to execute a turnover $B_i B_{i+1} \underline{G}_i = \tilde{\underline{G}}_{i+1} \hat{B}_i \hat{B}_{i+1}$. We now have $RG_i = P^T C^*(\tilde{\underline{G}}_{i+1} \hat{B} + \alpha\, e_1 \hat{\underline{y}}_1^T)P = P^T C^* \tilde{\underline{G}}_{i+1}(\hat{B} + \alpha\, e_1 \hat{\underline{y}}_1^T)P$. In the last step we have used the fact that $\tilde{G}_{i+1} e_1 = e_1$ because $i + 1 > 1$. To complete the procedure we just need to do one more turnover, in which $\tilde{\underline{G}}_{i+1}$ interacts with C_i^* and C_{i+1}^*. Specifically $C_{i+1}^* C_i^* \tilde{\underline{G}}_{i+1} = \hat{\underline{G}}_i \hat{C}_{i+1}^* \hat{C}_i^*$, resulting in $RG_i = P^T \hat{\underline{G}}_i \hat{C}^*(\hat{B} + \alpha\, e_1 \hat{\underline{y}}^T)P$. Finally we have $P^T \hat{\underline{G}}_i = \hat{G}_i P^T$, where $\hat{G}_i$ is the $n \times n$ version of $\hat{\underline{G}}_i$. Here it is important that $i \leq n-1$. The final result is $RG_i = \hat{G}_i P^T \hat{C}^*(\hat{B} + \alpha\, e_1 \hat{\underline{y}}^T)P = \hat{G}_i \hat{R}$.

The total computational effort required for the task is just two turnovers. The operation $\hat{\underline{y}}^T = \underline{y}^T \underline{G}_i$ is not actually performed, as we do not keep track of $\underline{y}$.

Pictorially for $n = 8$ and $i = 1$, we have

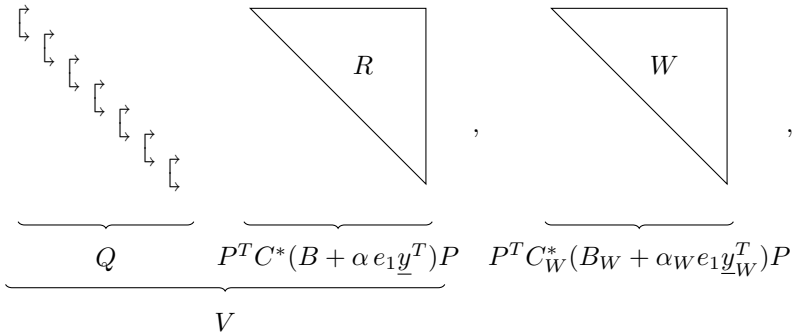


where we have not depicted P or P^T, and we have ignored the action on $\underline{y}^T$.

Notice that when we apply a core transformation G_{n-1}, we temporarily create $\tilde{\underline{G}}_n$, which makes use of row/column $n+1$. Here we see that the existence of an extra row/column is crucial to the functioning of the algorithm.

From left to right. We can pass a core transformation from left to right through R simply by reversing the above procedure. We omit the details

It is clear now that one can move a single core transformation through a factored upper triangular matrix in either direction by executing only two turnovers. From now on, to ease the notation, we will depict our Hessenberg-triangular pencil in a simpler format:



(3.6)

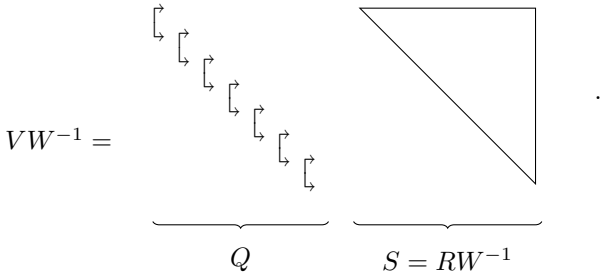
where we have replaced each upper triangular factor by a triangle. With this description, we can immediately apply the algorithms from Vandebril and Watkins [25]. For

completeness, however, we will redescribe the flow of a single QZ step.

4. The companion QZ algorithm. We have implemented both single-shift and double-shift companion QZ algorithms. For simplicity we will describe only the single-shift case, as the double-shift iteration is a straightforward extension; we refer to Aurentz et al. and Vandebril and Watkins [4,25]. The companion QZ algorithm is easily described by viewing it as a version of the companion QR algorithm applied to the matrix VW^{-1}. Clearly

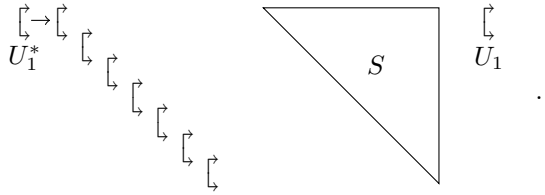
$$VW^{-1} = QRW^{-1} = QS,$$

where $S = RW^{-1}$ is upper triangular. Pictorially

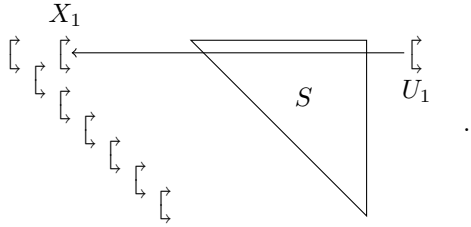


Of course we will find in the end that we do not have to invert W in practice.

To begin the iteration we select a suitable shift μ and compute $q = (V - \mu W)e_1$. Only the first two entries of q are nonzero, so we can construct a core transformation U_1 such that $U_1^* q = \alpha e_1$ for some α. Our first modification to VW^{-1} is to apply a similarity transformation by U_1:



We can immediately fuse U_1^* with Q_1 to make a new Q_1. (To keep the notation under control, we do not give the modified Q_1 a new name; we simply call it Q_1.) We can also pass U_1 through S to obtain



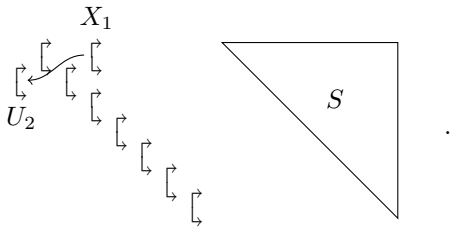
(4.1)

The details of passing a core transformation through S will be discussed below.

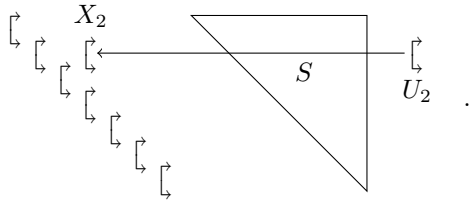
If we were to multiply the factors together, we would find that the matrix is no longer upper Hessenberg; there is a bulge in the Hessenberg form caused by a nonzero entry in position $(3, 1)$. The standard Francis algorithm chases the bulge until it

disappears off the bottom of the matrix. In our current setting we do not see a bulge. Instead we see an extra core transformation X_1 in (4.1), which is in fact the cause of the bulge. X_1 is the *misfit*. Instead of chasing the bulge, we will chase the misfit through the matrix until it disappears at the bottom. We therefore call this a *core chasing* algorithm.

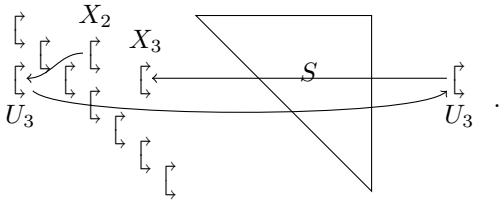
Proceeding from (4.1), the next step is to do a turnover $Q_1Q_2X_1 = U_2\hat{Q}_1\hat{Q}_2$. Core transformations $\hat{Q}_1$ and $\hat{Q}_2$ become the new Q_1 and Q_2. Pictorially



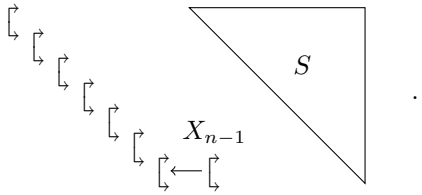
Next we do a similarity transformation, multiplying by U_2^* on the left and U_2 on the right. This has the effect of moving U_2 from the left side to the right side of the matrix. We can also pass U_2 through S to obtain



Now we are in the same position as we were at (4.1), except that the misfit has moved downward one position. The process continues as before:



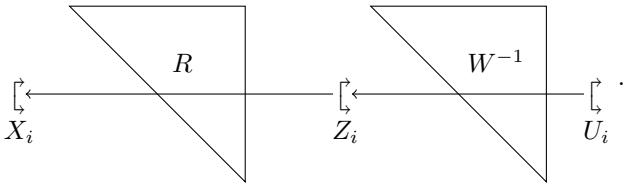
After $n-1$ such steps we arrive at



We can now fuse X_{n-1} with Q_{n-1}, completing the iteration.

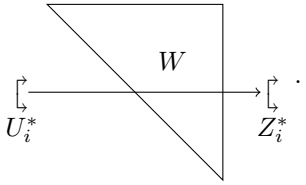
To complete the description, we now consider the details of passing a core trans-

formation through S. Since $S = RW^{-1}$, the operation has two stages:



(4.2)

Since we do not wish to invert W, we do not literally execute the operation depicted on the right of (4.2). Instead we do the equivalent operation



Since both R and W are unitary-plus-rank-one matrices stored in the factored form described in Section 3, each of the two stages costs two turnovers. Thus the computational cost of passing a core transformation through S is four turnovers.

The total cost of passing a core transformation through $VW^{-1} = QS$ is thus five turnovers. The corresponding cost for companion QR [4] is three turnovers, so we expect the companion QZ code to be slower than the companion QR code by a factor of 5/3. During a QR or QZ iteration the misfit gets passed through the matrix about n times, so the total number of turnovers is $5n$ for a QZ step and $3n$ for a QR step. Either way the flop count is $O(n)$. Reckoning $O(n)$ total iterations, we get a flop count of $O(n^2)$ for both companion QR and QZ, with companion QZ expected to be slower by a factor of 5/3.

In Figure 4.1 we show execution times for our companion QR and QZ codes on polynomials of degree from 4 up to about 16000. We also make a comparison with the code from Boito, Eidelman, and Gemignani (BEGQZ) [8] and the LAPACK QR and QZ codes ZHSEQR and ZHGEQZ. Straight lines indicating $O(n^2)$ and $O(n^3)$ performance are included for comparison purposes. Our codes are the fastest. The lower plot in the figure corroborates our expectation that companion QZ will be slower than companion QR by a factor of about 5/3. For polynomials of low degree the LAPACK QR code DHSEQR is roughly as fast as our codes, but from degree 100 or so we are much faster than all other methods. The execution time curves for our companion QR and QZ codes are about parallel to the $O(n^2)$ line, indicating $O(n^2)$ execution time. The same is true of the BEGQZ method.

Table 4.1 shows the execution times for a few selected high degrees.

For this experiment we used a single core of an Intel® Xeon® CPU E5-2697 v3 running at 2.60GHz with 35 MB shared cache and 128 GB RAM. We used the Gnu Compiler Collection gcc version 4.8.5 on an Ubuntu 14.04.1. For the comparison we used Lapack version 3.7.1.

5. Backward Error Analysis. The norm symbol $\|\cdot\|$ will denote the 2-norm, i.e. the Euclidean norm for vectors and the spectral norm for matrices. These are the norms that we use in our backward error analysis. In our numerical tests we use the Frobenius matrix norm $\|\cdot\|_F$ instead of the spectral norm because it is easier to compute. In fact the choice of norms is not important to the analysis; we could use other common norms such as $\|\cdot\|_1$ or $\|\cdot\|_\infty$ with no essential change in the results.

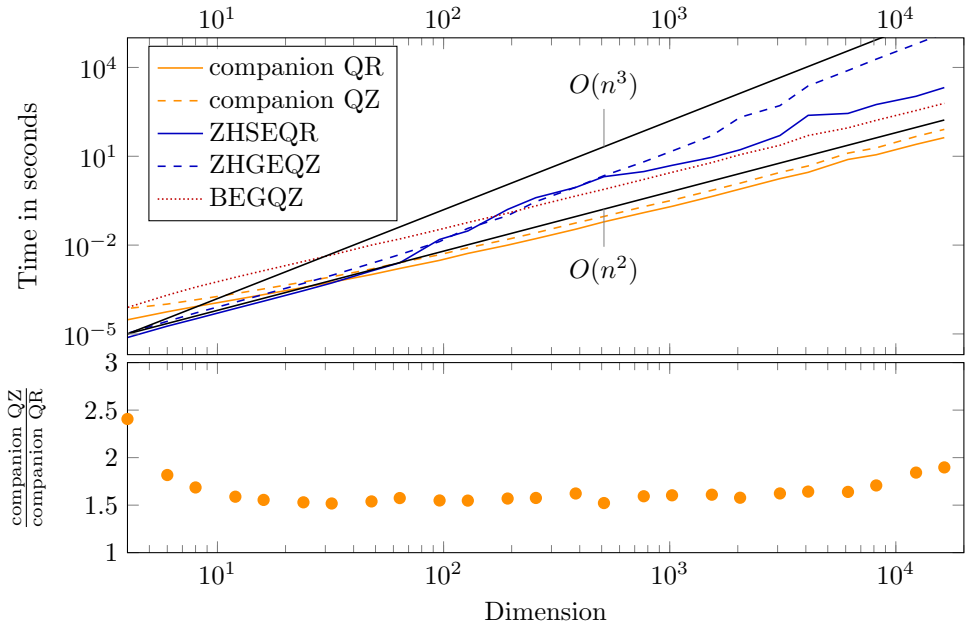


FIG. 4.1. *Execution times for several methods on polynomials of degree from 4 up to 16384. $O(n^2)$ and $O(n^3)$ lines are included for comparison.*

TABLE 4.1
Execution times in seconds for selected high degrees

degree	3072	6144	12288
comp. QR	2	8	25
comp. QZ	3	13	47
BEGQZ	23	91	350
ZHSEQR	50	280	1062
ZHGEQZ	514	7885	61856

In addition we use the following conventions: $\doteq$ denotes an equality where second and higher order terms are dropped, $\lesssim$ stands for less than or equal up to a modest multiplicative constant typically depending on n as a low-degree polynomial, $\approx$ denotes equal up to a modest multiplicative constant. The symbol u denotes the *unit roundoff*, which is about 10^{-16} for IEEE binary64 arithmetic.

Van Dooren and Dewilde [24] were the first to investigate the backward stability of polynomial root finding via companion matrices and pencils. Edelman and Murakami [14] revisited this analysis, focusing on scalar polynomials. Jónsson and Vavasis [17] presented a clear summary of these results.

There are two important measures of backward accuracy when dealing with companions: the backward error (i) on the companion matrix or pencil and (ii) on the coefficients of the original polynomial. Let $a = \begin{bmatrix} a_0 & \cdots & a_n \end{bmatrix}^T$, the coefficient vector of p. Van Dooren and Dewilde [24] showed that pushing an unstructured error further back from the pencil (or matrix) to the polynomial coefficients introduces an additional factor $\|a\|$ in the backward error. We will show that we can do better since

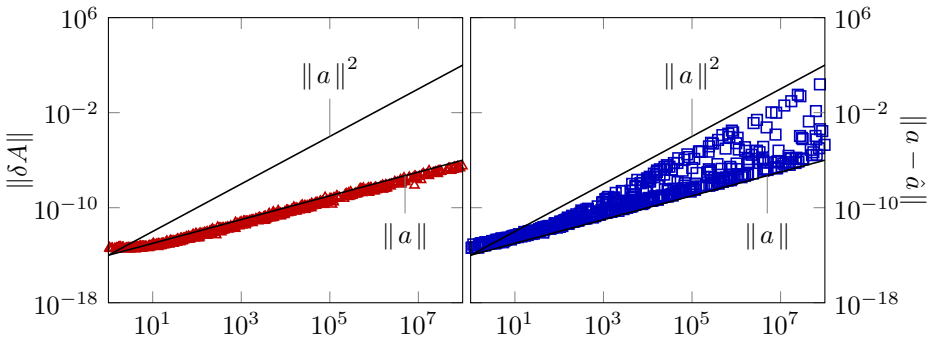


FIG. 5.1. *Backward error on the companion matrix (left) and the coefficient vector (right) as a function of* $\|a\|$ *when roots are computed by unstructured LAPACK code.*

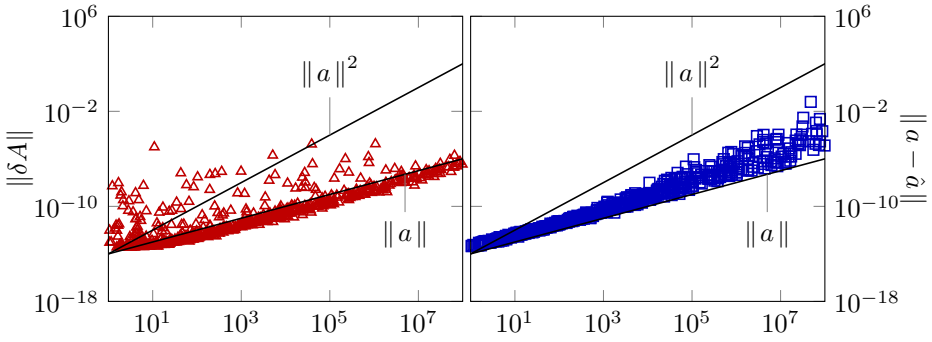


FIG. 5.2. *Same experiment as in Figure 5.1, except that the matrix is balanced before the eigenvalue computation.*

the backward error produced by our companion QR method is highly structured.

To illustrate what happens in the unstructured case we show in Figure 5.1 the absolute backward error as a function of $\|a\|$ when the companion matrix eigenvalue problem is solved by the QR algorithm from LAPACK. Eight hundred random polynomials with varying norm between 1 and 10^8 are produced by a method described below. For each sample we plot a single point: the backward error against $\|a\|$. Black lines with slopes corresponding to $\|a\|$ and $\|a\|^2$ performance are also provided.

The graph on the left is the backward error on the companion matrix A. We see that this grows linearly as a function of $\|a\|$. This is consistent with the backward stability of the QR algorithm, which guarantees that the computed eigenvalues are the exact eigenvalues of a slightly perturbed matrix $A+\delta A$ with $\|\delta A\| \lesssim u\,\|A\| \approx u\,\|a\|$.

The graph on the right shows the backward error on the coefficients of the polynomial as a function of $\|a\|$. Note that the growth is quadratic in $\|a\|$ in the worst cases, consistent with the analysis of Van Dooren and Dewilde [24] and Edelman and Murakami [14].

In this and all subsequent experiments we produced polynomials with increasing coefficient norm, parametrized by an integer $\rho = 1, \ldots, 8$. (Exception: In Figure 5.7 we let ρ go up to 20.) For each ρ we ran 100 experiments. We chose polynomials of degree 50, but the results for degree 6, 20, and 200 were very similar. For each of the 51 coefficients of each polynomial we choose three random numbers ν, μ, and η,

uniformly distributed in $[0,1]$. The coefficient is a complex number with argument $2\pi\nu$ and absolute value $(2\mu-1)10^{\rho(2\eta-1)}$. (Polynomials of similar type have been used in, e.g., [12].)

Balancing [22] is often touted as an important step in solving the companion eigenvalue problem. In Figure 5.2 we have repeated the same experiment as in Figure 5.1, except that a balancing step is added. We see that, at least for the class of polynomials we considered, balancing does not help much and certainly does not change our conclusions.

5.1. Backward error on the companion matrix. We begin by considering the backward error of our structured companion matrix code (companion QR) [4]. We will fix the error in the analysis of [4] and make other substantial improvements. We will take p to be monic with coefficient vector $a = \begin{bmatrix} a_0 & \cdots & a_{n-1} & 1 \end{bmatrix}^T$. The companion matrix is

$$A = \begin{bmatrix} & & & -a_0 \\ 1 & & & -a_1 \\ & \ddots & & \vdots \\ & & 1 & -a_{n-1} \end{bmatrix}. \tag{5.1}$$

Clearly $1 \leq \|a\| \approx \|A\|$.

When we run the companion QR algorithm on A, we transform it to U^*AU, where U is the product of all of the core transformations that took part in similarity transformations. At the same time Q and R are transformed to U^*QX and X^*RU, where X is the product of all core transformations that were ejected from R and absorbed by Q. (Similarly we can view U as the product of all core transformations that were ejected from Q and absorbed by R.)

In floating-point arithmetic we have

$$\hat{A} = U^*(A + \delta A)U.$$

where δA is the backward error. The roots that we compute are exactly the eigenvalues of $\hat{A}$ and of $A + \delta A$. We would like to get a bound on $\|\delta A\|$.

We begin by looking at the backward error on R, and to this end we consider first the larger matrix $\underline{R}$ (3.2), which we can write in the factored form (3.3).

$$\hat{\underline{R}} = \underline{X}^*(\underline{R} + \underline{\delta R})\underline{U},$$

where $\underline{U} = \mathrm{diag}\{U, 1\}$ and $\underline{X} = \mathrm{diag}\{X, 1\}$. Consider the unitary and rank-one parts separately: $\underline{R} = \underline{R}_u + \underline{R}_o$, where $\underline{R}_u = C^*B$ and $\underline{R}_o = \alpha\, C^* e_1 \underline{y}^T = \alpha\, \underline{x}\, \underline{y}^T$. Here we have introduced a new symbol $\underline{x} = C^* e_1 = \alpha^{-1}\underline{z}$. We note that $\|\underline{x}\| = \|\underline{y}\| = 1$, and $|\alpha| = \|\underline{z}\| = \|a\|$ in this (monic) case. We will determine backward errors associated with these two parts: $\underline{\delta R} = \underline{\delta R}_u + \underline{\delta R}_o$.

Since $\underline{R}_u = C^*B$, we have $\underline{R}_u + \underline{\delta R}_u = (C + \delta C)^*(B + \delta B)$. In [4] we noted that $\|\delta B\| \lesssim u$ and $\|\delta C\| \lesssim u$. (This is because the operations that take place on B and C amount to matrix multiplications and QR decompositions (on unitary matrices!) and are backward stable.) We deduce that $\underline{\delta R}_u \doteq \delta C^* B + C^* \delta B$, and $\|\underline{\delta R}_u\| \lesssim u$.

For the rank-one part recall from (3.4) that

$$\alpha \underline{y}^T = -\rho^{-1} e_{n+1}^T \underline{R}_u, \quad \text{where} \quad \rho = e_{n+1}^T C^* e_1.$$

The backward error in ρ is given by

$$\rho + \delta\rho = e_{n+1}^T (C + \delta C)^* e_1,$$

so $\delta\rho = e_{n+1}^T \delta C^* e_1$, and $|\delta\rho| \lesssim u$. (In exact arithmetic ρ is invariant under Francis iterations. Our mistake in [4] was to treat it as a constant in the backward error analysis.)

An easy computation shows that $-\rho^{-1} = \alpha$: Indeed, $\rho = e_{n+1}^T C^* e_1 = e_{n+1}^T \underline{x} = x_{n+1} = \alpha^{-1} z_{n+1} = -\alpha^{-1}$. Thus $\underline{y}^T = e_{n+1}^T \underline{R}_u$ and

$$\underline{R}_o = \alpha\, \underline{x}\, \underline{y}^T = \alpha\, C^* e_1 e_{n+1}^T \underline{R}_u.$$

Noting that $-(\rho + \delta\rho)^{-1} = (1 - \alpha\,\delta\rho)^{-1}\alpha$, and defining $\beta = (1 - \alpha\,\delta\rho)^{-1}$, we see that the backward error in $\underline{R}_o$ is given by

$$\underline{R}_o + \underline{\delta R}_o = \beta\,\alpha (C + \delta C)^* e_1 e_{n+1}^T (\underline{R}_u + \underline{\delta R}_u).$$

Define $\underline{\delta x} = \delta C^* e_1$ and $\underline{\delta y}^T = e_{n+1}^T \underline{\delta R}_u$, and note that $\|\underline{\delta x}\| \lesssim u$ and $\|\underline{\delta y}\| \lesssim u$. Then

$$\underline{R}_o + \underline{\delta R}_o = \beta\,\alpha (\underline{x} + \underline{\delta x})(\underline{y} + \underline{\delta y})^T. \tag{5.2}$$

This should be compared with the unperturbed equation $\underline{R}_o = \alpha\, \underline{x}\, \underline{y}^T$. Note the extra factor β in (5.2).

Recall that $|\alpha| = \|a\| = \|\underline{z}\| = \|\underline{R}_o\| \approx \|R\| = \|A\|$, and the approximation is excellent when $\|A\|$ is large. Thus we will use the approximation $|\alpha| = \|a\| \approx \|A\|$ without further comment.

We have proved the following lemma.

LEMMA 5.1. *Assume $\|a\| \ll 1/u$, where u is the unit roundoff, and let $\beta = (1 - \alpha\,\delta\rho)^{-1} > 0$. Then*

$$\hat{\underline{R}} = \underline{X}^* (\underline{R} + \underline{\delta R}) \underline{U},$$

where

$$\underline{R} + \underline{\delta R} = \underline{R}_u + \underline{\delta R}_u + \beta\,\alpha\,(\underline{x} + \underline{\delta x})(\underline{y} + \underline{\delta y})^T$$

with $\|\underline{\delta R}_u\| \lesssim u$, $\|\underline{\delta x}\| \lesssim u$, and $\|\underline{\delta y}\| \lesssim u$.

The assumption $\|a\| \ll 1/u$ is not very restrictive at all, and we have barely used it here. This just serves to guarantee that $1 - \alpha\,\delta\rho$ is greater than zero and therefore has an inverse. We will make more substantial use of this assumption in Theorem 5.5 and in general whenever we drop a term of second order. If we have two terms g and h satisfying $g \lesssim \|a\|\,u$ and $h \lesssim u$, then gh is negligible because $gh \lesssim \|a\|\,u^2 \approx u$.

For our first theorem we require the more restrictive assumption $\|a\| \lesssim 1/\sqrt{u}$, which implies $|\alpha\,\delta\rho| \lesssim \sqrt{u}$ and therefore allows us to make the approximation

$$\beta = (1 - \alpha\,\delta\rho)^{-1} \doteq 1 + \alpha\,\delta\rho.$$

THEOREM 5.2. *Assume $\|a\| \lesssim 1/\sqrt{u}$. If the companion eigenvalue problem is solved by our companion QR code, then*

(a) $\hat{A} = U^*(A + \delta A)U$, where $\|\delta A\| \lesssim u\|A\|^2$.

(b) $\hat{Q} = U^*(Q + \delta Q)X$, where $\|\delta Q\| \lesssim u$.

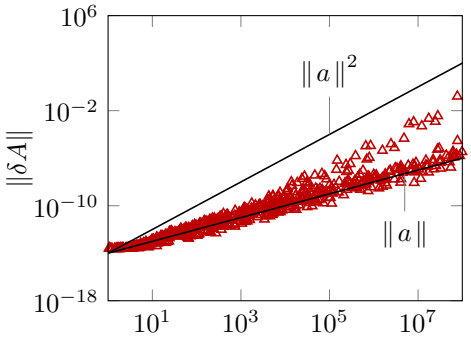


FIG. 5.3. *Companion QR method; norm of the backward error on the companion matrix A plotted against $\|a\|$*

(c) $\hat{R} = X^*(R + \delta R)U$, *where* $\|\delta R\| \lesssim u\|A\|^2$.
(d) More precisely,

$$\delta R = \delta R_u + \alpha^2 \delta\rho\, x\, y^T + \alpha(\delta x\, y^T + x\, \delta y^T),$$

where $\|\delta R_u\| \lesssim u$, $|\delta\rho| \lesssim u$, $\|\delta x\| \lesssim u$, *and* $\|\delta y\| \lesssim u$.

Remark 5.3. The square on the norm of A in the result $\|\delta A\| \lesssim u\|A\|^2$ is cause for concern, as it suggests that the backward error will be especially bad when $\|A\|$ is large. We will offer a solution to this problem in the next theorem.

Proof. Everything follows from part (d). The four terms in δR shown there are not all of the same size. One term has a factor α^2, and this is the big one, as the others only have a factor α^1 or α^0. Because of this big term, $\|\delta R\| \lesssim u\,|\alpha|^2 \approx u\,\|A\|^2$. Thus (c) follows from (d). Part (b) was established earlier (cf. discussion of $B + \delta B$ and $C + \delta C$ above). Part (a) follows from (b) and (c) because $\delta A \doteq \delta Q\, R + Q\, \delta R$.

Now we just need to prove (d). Using the approximation $\beta \doteq 1 + \alpha\,\delta\rho$, in (5.2) we get

$$\begin{aligned} \underline{R}_o + \underline{\delta R}_o &\doteq (1 + \alpha\,\delta\rho)\,\alpha(\underline{x} + \underline{\delta x})(\underline{y} + \underline{\delta y})^T \\ &\doteq \underline{R}_o + \alpha\,\delta\rho\,\alpha\,\underline{x}\underline{y}^T + \alpha(\underline{\delta x}\underline{y}^T + \underline{x}\underline{\delta y}^T), \end{aligned}$$

so

$$\delta R_o = P^T \underline{\delta R}_o P \doteq \alpha^2 \delta\rho\, x\, y^T + \alpha(\delta x\, y^T + x\, \delta y^T),$$

where $\delta x = P^T \underline{\delta x}$ and so on. This establishes (d). ∎

Example 5.4. As a numerical test of Theorem 5.2 we computed the backward error on the companion matrix as follows. During the companion QR iterations we accumulated the transforming matrix U. We then computed $\check{A} = U\hat{A}U^*$. The backward error on A is $\delta A = \check{A} - A$. Figure 5.3 shows that the backward error on A grows in proportion to $\|A\|^2$ in the worst case.

The dependence on $\|A\|^2$ is annoying. We can get rid of it by distributing the backward error on the pencil $\lambda I - A$ instead of pushing it all onto the matrix A.

THEOREM 5.5. *Let A be as in Theorem 5.2, but assume the less stringent condition $\|a\| \ll 1/u$. Then the eigenvalues of A computed by the companion QR algorithm are exactly the eigenvalues of a perturbed pencil $\lambda\,(I+\delta I)-(A+\delta A)$ with $\|\delta I\| \lesssim u\|A\|$ and $\|\delta A\| \lesssim u\|A\|$.*

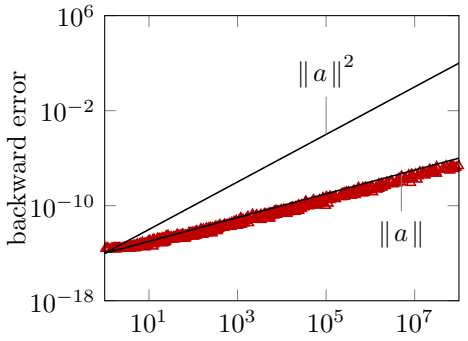


FIG. 5.4. *Backward error of companion QR on the pencil $\lambda I - A$*

Proof. We will use Lemma 5.1 instead of Theorem 5.2, so we do not need the more stringent assumption on $\|a\|$.

The big part of the backward error is in the rank-one part, so let's start there. By (5.2)

$$\underline{R}_o + \underline{\delta R}_o = \beta(\underline{R}_o + \alpha\,\delta E),$$

where $\delta E \doteq \underline{\delta x}\,\underline{y}^T + \underline{x}\,\underline{\delta y}^T$ and satisfies $\|\delta E\| \lesssim u$. Our simple plan is to multiply the entire pencil by the factor $\beta^{-1} = 1 - \alpha\,\delta\rho$ to get rid of the big part of the error.

The backward error on A, which was denoted δA up to this point, will now be denoted $\widetilde{\delta A}$. Thus the pencil to which we refer is now denoted $\lambda I - (A + \widetilde{\delta A})$. We will multiply it by β^{-1} to get

$$\lambda(I + \delta I) - (A + \delta A) = \beta^{-1}(\lambda I - (A + \widetilde{\delta A})).$$

The λI part is easy to deal with: $\beta^{-1} I = (1 - \alpha\,\delta\rho)I = I + \delta I$, where $\delta I = -\alpha\,\delta\rho I$, and $\|\delta I\| \lesssim u\,\|A\|$.

We have to be a bit more careful with $\beta^{-1}(A + \widetilde{\delta A})$.

$$\begin{aligned} A + \delta A &= \beta^{-1}(A + \widetilde{\delta A}) \\ &= \beta^{-1}(Q + \delta Q)P^T(\underline{R}_u + \underline{\delta R}_u + \beta[\underline{R}_o + \alpha\,\delta E])P \\ &= (Q + \delta Q)P^T((1 - \alpha\,\delta\rho)(\underline{R}_u + \underline{\delta R}_u) + \underline{R}_o + \alpha\,\delta E)P \\ &\doteq (Q + \delta Q)P^T(\underline{R}_u - \alpha\,\delta\rho\,\underline{R}_u + \underline{\delta R}_u + \underline{R}_o + \alpha\,\delta E)P. \end{aligned}$$

In the last step we have discarded the term $\alpha\,\delta\rho\,\underline{\delta R}_u$, which is negligible because the condition $\|a\| \ll 1/u$ implies $|\alpha|\,|\delta\rho|\,\|\underline{\delta R}_u\| \lesssim \|a\|\,u^2 \approx u$. Thus

$$\delta A \doteq \delta Q\,R + QP^T(-\alpha\,\delta\rho\,\underline{R}_u + \underline{\delta R}_u + \alpha\,\delta E)P.$$

It follows that $\|\delta A\| \lesssim u\|A\|$. ∎

Example 5.6. As a numerical test of Theorem 5.5 we computed the Backward error δA as in Example 5.4. We then computed γ such that the pencil $\gamma(\lambda I - (A + \delta A))$ is as close as possible to the pencil $\lambda I - A$ in the sense that $e_P = \|\gamma[I\ A + \delta A] - [I\ A]\|_F$ is minimized. Figure 5.4 plots e_P against $\|a\|$. We see that this backward error grows at most linearly in $\|a\|$.

5.2. Backward error on the polynomial coefficients. We continue to study the backward error of the companion QR algorithm on the companion matrix (5.1), leaving the companion pencil case for later. The monic polynomial

$$p(\lambda) = a_0 + a_1\lambda + a_2\lambda^2 + \cdots + a_{n-1}\lambda^{n-1} + \lambda^n$$

is associated with the coefficient vector

$$a = \begin{bmatrix} a_0 & \cdots & a_{n-1} & 1 \end{bmatrix}^T \in \mathbb{C}^{n+1}.$$

When we compute the zeros of p, we don't get the exact zeros, but we hope that they are the zeros of a "nearby" polynomial. Let $\lambda_1, \ldots, \lambda_n$ denote the computed zeros and

$$\tilde{p}(\lambda) = (\lambda - \lambda_1)\cdots(\lambda - \lambda_n).$$

This is the monic polynomial that has the computed roots as its zeros. We can compute the coefficients of $\tilde{p}$:

$$\tilde{p}(\lambda) = \tilde{a}_0 + \tilde{a}_1\lambda + \tilde{a}_2\lambda^2 + \cdots + \tilde{a}_{n-1}\lambda^{n-1} + \lambda^n$$

and the corresponding coefficient vector

$$\tilde{a} = \begin{bmatrix} \tilde{a}_0 & \cdots & \tilde{a}_{n-1} & 1 \end{bmatrix}^T \in \mathbb{C}^{n+1}.$$

This computation is done in multiple precision arithmetic using MPFUN [21]. The quantity $\delta a = \tilde{a} - a$ is the backward error on the coefficients. We would like to show that $\|\,\delta a\,\|$ is tiny.

Theorem 5.2 shows that the norm of the backward error on the companion matrix A depends on the square of the norm of the matrix. According to Van Dooren and Dewilde [24] and Edelman and Murakami [14], the backward error on the polynomial coefficients will then depend on the cube of the norm, that is, $\|\,\delta a\,\| \lesssim u\,\|\,a\,\|^3$. In this section we show that we can do better: By exploiting the structure of the problem, we can make an argument that depends only trivially on the analyses of [14, 24] and shows that the backward error on the polynomial depends quadratically on the norm: $\|\,\delta a\,\| \lesssim u\,\|\,a\,\|^2$. This is under the assumption that the approximating polynomial is monic. If we allow a non-monic approximation $\hat{p} = \gamma\tilde{p}$ (similar to what we did in Theorem 5.5), the backward error on the polynomial depends linearly on the norm, that is, $\|\,\delta a\,\| \lesssim u\,\|\,a\,\|$. This is an optimal result and is better than what is achieved by LAPACK's QR algorithm or Matlab's `roots` command.

Now let's get started on the analysis. In the factorization $A = QR$, the triangular factor can be written as $R = I + (\alpha\,x - e_n)y^T$, using notation established earlier in this section. Thus

$$\lambda I - A = (\lambda I - Q) - Q(\alpha\,x - e_n)y^T. \tag{5.3}$$

In order to exploit this equation we need the following known result [16, p. 26]. For the reader's convenience we sketch a proof.

LEMMA 5.7. *Let $K \in \mathbb{C}^{n\times n}$, and let w, $v \in \mathbb{C}^n$. Then*

$$\det(K + w\,v^T) = \det(K) + v^T\,\mathrm{adj}(K)w,$$

where $\mathrm{adj}(K)$ *denotes the adjugate matrix of* K.

Proof. We will prove the result for nonsingular K; it then follows for all K by continuity. The special case $\det(I + u\,v^T) = 1 + v^T u$ is easily verified. Thus $\det(K + w\,v^T) = \det(K)\det(I + K^{-1}w\,v^T) = \det(K)(1 + v^T K^{-1} w) = \det K + v^T \operatorname{adj}(K) w$, as $\operatorname{adj}(K) = \det(K)\,K^{-1}$ when K is nonsingular. □

COROLLARY 5.8. $\det(\lambda I - A) = \det(\lambda I - Q) - y^T \operatorname{adj}(\lambda I - Q) Q(\alpha\, x - e_n)$.

Proof. Apply Lemma 5.7 to (5.3). □

The entries of the adjugate matrix are determinants of order $n-1$, so $\operatorname{adj}(\lambda I - Q)$ is a matrix polynomial of degree $n-1$:

$$\operatorname{adj}(\lambda I - Q) = \sum_{k=0}^{n-1} G_k \lambda^k. \tag{5.4}$$

Since also $\det(\lambda I - Q) = \lambda^n - 1$, we can write the characteristic polynomial of A as

$$p(\lambda) = \det(\lambda I - A) = \lambda^n - 1 - \sum_{k=0}^{n-1} y^T G_k Q(\alpha\, x - e_n)\lambda^k. \tag{5.5}$$

Thus the coefficients of p are

$$a_k = q_k - y^T G_k Q(\alpha\, x - e_n), \qquad k = 0, \ldots, n,$$

where $G_n = 0$, $q_n = 1$, $q_0 = -1$, and $q_k = 0$ otherwise.

The roots that we actually compute are the zeros of a perturbed polynomial

$$\tilde{p}(\lambda) = \det(\lambda I - (A + \delta A)) = \sum_{k=0}^{n} (a_k + \delta a_k)\lambda^k.$$

Our plan now is to use (5.5) to determine the effect of the perturbation δA on the coefficients of the characteristic polynomial. That is, we want bounds on $|\,\delta a_k\,|$.

LEMMA 5.9. *Assume* $\|\,a\,\| \ll 1/u$, *and let* $\beta = (1 - \alpha\,\delta\rho)^{-1}$. *Then*

$$R + \delta R = (I + \delta I) + \beta\,(\alpha(x + \delta x) - e_n)(y + \delta y)^T,$$

where $\|\,\delta I\,\| \lesssim u$, $\|\,\delta x\,\| \lesssim u$, *and* $\|\,\delta y\,\| \lesssim u$.

This should be compared with the unperturbed equation $R = I + (\alpha\, x - e_n)y^T$. Again note the extra factor β.

Proof. From Lemma 5.1 we know that

$$\underline{R} + \underline{\delta R} = \underline{R}_u + \underline{\delta R}_u + \beta\,\alpha\,(\underline{x} + \underline{\delta x})(\underline{y} + \underline{\delta y})^T.$$

Projecting this down to $n \times n$ matrices and giving δx the new name $\widetilde{\delta x}$, we get

$$R + \delta R = R_u + \delta R_u + \beta\,\alpha\,(x + \widetilde{\delta x})(y + \delta y)^T, \tag{5.6}$$

with $\|\,\delta R_u\,\| \lesssim u$, $\|\,\widetilde{\delta x}\,\| \lesssim u$, and $\|\,\delta y\,\| \lesssim u$.

Recalling the form of $\underline{R}_u$, we see that $R_u = I - e_n y^T$, so

$$\begin{aligned} R_u + \delta R_u &= I - e_n y^T + \delta R_u \\ &= I - e_n(y + \delta y)^T + e_n\,\delta y^T + \delta R_u \\ &= I + \delta I - e_n(y + \delta y)^T, \end{aligned} \tag{5.7}$$

where $\delta I = e_n \delta y^T + \delta R_u$, so $\|\delta I\| \lesssim u$. Now, substituting (5.7) into (5.6) and using $\beta(1 - \alpha\,\delta\rho) = 1$, we get

$$\begin{aligned} R + \delta R &= I + \delta I - \beta(1 - \alpha\,\delta\rho)e_n(y + \delta y)^T + \beta\,\alpha(x + \widetilde{\delta x})(y + \delta y)^T \\ &= I + \delta I + \beta[-e_n + \alpha\,\delta\rho\,e_n + \alpha(x + \widetilde{\delta x})](y + \delta y)^T \\ &= I + \delta I + \beta(\alpha(x + \delta x) - e_n)(y + \delta y)^T, \end{aligned}$$

where $\delta x = \widetilde{\delta x} + \delta\rho\, e_n$ and $\|\delta x\| \lesssim u$. This completes the proof. ∎

For our next lemma we need some facts about adjugates. Stewart [23] showed that if the singular values of B are $\sigma_1 \geq \ldots \geq \sigma_n$, then the singular values of $\mathrm{adj}(B)$ are $\gamma_1, \ldots, \gamma_n$, where

$$\gamma_j = \prod_{i \neq j} \sigma_i.$$

This is easily proved by considering the relationship between the singular value decompositions of B and its adjugate. If follows that $\|\mathrm{adj}(B)\| = \sigma_1 \cdots \sigma_{n-1}$. We will use a perturbation result from [23]: If $F = \mathrm{adj}(B + E) - \mathrm{adj}(B)$, then

$$\|F\| \lesssim \sigma_1\sigma_2 \cdots \sigma_{n-2}\|E\|. \tag{5.8}$$

This is from the second displayed inequality on page 156 of [23].

In the next lemma we will apply (5.8) to the matrices $Z_j = \mathrm{adj}(\xi^j I - Q)$, where $\xi = e^{-2\pi i/n}$. Recall that the eigenvalues of Q are ξ^0, ξ^1, ξ^2, $\ldots$, ξ^{n-1}, so the eigenvalues of $\xi^j I - Q$ are $\xi^j - \xi^0, \ldots, \xi^j - \xi^{n-1}$. Since $\xi^j I - Q$ is normal, its singular values are $|\xi^j - \xi^0|, \ldots, |\xi^j - \xi^{n-1}|$. Clearly one of these is zero, so we have $\sigma_n = 0$. The next smallest one is equal to the distance between two adjacent nth roots of unity, so

$$\sigma_{n-1} \geq 4/n.$$

To make use of (5.8) we need an estimate of $\sigma_1 \cdots \sigma_{n-2}$, which can be written as

$$\sigma_1 \cdots \sigma_{n-2} = \frac{\sigma_1 \cdots \sigma_{n-1}}{\sigma_{n-1}} = \frac{\|Z_j\|}{\sigma_{n-1}}.$$

We have a lower bound for the denominator. Now, what about the numerator? By symmetry $\|Z_j\|$ is independent of j, so we consider $Z_0 = \mathrm{adj}(I - Q)$. The $n - 1$ nonzero singular values of $I - Q$ are $|1 - \xi^1|, \ldots, |1 - \xi^{n-1}|$, so

$$\|Z_j\| = \|Z_0\| = \prod_{k=1}^{n-1} |1 - \xi^k|.$$

Define a polynomial f of degree $n - 1$ by

$$f(\lambda) = \prod_{i=1}^{n-1} (\lambda - \xi^i) = \frac{\lambda^n - 1}{\lambda - 1} = \lambda^{n-1} + \lambda^{n-2} + \cdots + \lambda + 1.$$

Then

$$\|Z_j\| = |f(1)| = f(1) = n.$$

Thus the matrices Z_j all satisfy

$$\sigma_1 \cdots \sigma_{n-2} \le \frac{n^2}{4} \qquad \text{(and therefore } \sigma_1 \cdots \sigma_{n-2} \lesssim 1\text{)}. \tag{5.9}$$

LEMMA 5.10. *If* $\|\,\delta I\,\| \lesssim \epsilon$ *and* $\|\,\delta Q\,\| \lesssim \epsilon$ *for some* $\epsilon > 0$, *then*

$$\operatorname{adj}(\lambda(I+\delta I) - (Q+\delta Q)) = \sum_{k=0}^{n-1} (G_k + \delta G_k)\lambda^k \tag{5.10}$$

with $\|\,\delta G_k\,\| \lesssim \epsilon$, $k = 0, \ldots, n-1$.

Proof. Let $Z(\lambda) = \operatorname{adj}(\lambda I - Q)$ and $\xi = e^{-2\pi i/n}$. Setting $\lambda = \xi^j$ and letting $Z_j = Z(\xi^j)$, we have

$$Z_j = \sum_{k=0}^{n-1} G_k \xi^{jk}, \qquad j = 0, \ldots, n-1.$$

This is a discrete Fourier transform, and its inverse is

$$G_k = \frac{1}{n} \sum_{j=0}^{n-1} Z_j \xi^{-jk}, \qquad k = 0, \ldots, n-1.$$

We can also consider the discrete Fourier transform of the perturbed quantity $\operatorname{adj}(\lambda(I+\delta I) - (Q+\delta Q))$:

$$Z_j + \delta Z_j = \sum_{k=0}^{n-1} (G_k + \delta G_k)\xi^{jk},$$

$$G_k + \delta G_k = \frac{1}{n} \sum_{j=0}^{n-1} (Z_j + \delta Z_j)\xi^{-jk},$$

and hence

$$\delta G_k = \frac{1}{n} \sum_{j=0}^{n-1} \delta Z_j\, \xi^{-jk},$$

and

$$\|\,\delta G_k\,\| \le \frac{1}{n} \sum_{j=0}^{n-1} \|\,\delta Z_j\,\|, \qquad k = 0, \ldots, n-1.$$

This shows that it suffices to prove that $\|\,\delta Z_j\,\| \lesssim \epsilon$ for $j = 0, \ldots, n-1$. This is now easy because of our preparations. From (5.10) we see that

$$\delta Z_j = \operatorname{adj}(\xi^j(I+\delta I) - (Q+\delta Q)) - \operatorname{adj}(\xi^j I - Q).$$

We can apply (5.8) with $B = \xi^j I - Q$ and $E = \xi^j \delta I - \delta Q$ and use (5.9) to deduce that $\|\,\delta Z_j\,\| \lesssim \|\,E\,\| \lesssim \epsilon$. ∎

THEOREM 5.11. *Suppose we apply the companion QR algorithm to the monic polynomial* p *with coefficient vector* a. *Let* $\tilde{p}$, *with coefficient vector* $\tilde{a} = a + \delta a$, *denote*

the monic polynomial that has the computed roots as its exact zeros. If $\|a\| \lesssim 1/\sqrt{u}$, *then*

$$\|\delta a\| \lesssim u\|a\|^2.$$

Proof.
Using Lemma 5.9, and giving δQ the new name $\widetilde{\delta Q}$, we get

$$\begin{aligned} A+\delta A &= (Q+\widetilde{\delta Q})(R+\delta R) \\ &= (Q+\widetilde{\delta Q})[(I+\delta I)+\beta\,(\alpha(x+\delta x)-e_n)(y+\delta y)^T] \\ &= (Q+\delta Q)+\beta\,(Q+\widetilde{\delta Q})(\alpha(x+\delta x)-e_n)(y+\delta y)^T, \end{aligned}$$

where $\delta Q \doteq Q\,\delta I+\widetilde{\delta Q}$ and $\|\delta Q\| \lesssim u$. Applying Lemma 5.10 with $\epsilon = u$ (and $\delta I = 0$) we deduce that (5.10) holds with $\|\delta G_k\| \lesssim u$. Now, using (5.5) with A replaced by $A+\delta A$, we get

$$\begin{aligned} \tilde{p}(\lambda) &= \det(\lambda I-(A+\delta A)) \\ &= \det(\lambda I-(Q+\delta Q)) \\ &\quad +\beta\sum_{k=0}^{n-1}(y+\delta y)^T(G_k+\delta G_k)(Q+\widetilde{\delta Q})(\alpha\,(x+\delta x)-e_n)\lambda^k. \end{aligned} \tag{5.11}$$

The perturbation in the Q part is insignificant: $\det(\lambda I-Q) = \sum_{k=0}^{n} q_k\lambda^k$, where $q_n = 1$, $q_0 = -1$, and $q_k = 0$ otherwise. Writing

$$\det(\lambda I-(Q+\delta Q)) = \sum_{k=0}^{n}(q_k+\delta q_k)\lambda^k$$

and invoking [14,24], we have $|\,\delta q_k\,| \lesssim u\,\|Q\|^2$ because $\|\delta Q\| \lesssim u$. Since $\|Q\| = 1$, we have $|\,\delta q_k\,| \lesssim u$.

Nothing up to this point depends on the assumption $\|a\| \lesssim 1/\sqrt{u}$. We now use this assumption so that we can make the approximation $\beta \doteq 1+\alpha\,\delta\rho$ in (5.11). Expanding this and ignoring higher order terms, we obtain

$$\begin{aligned} \delta a_k \doteq\ & \delta q_k+\alpha\,\delta\rho\,y^TG_kQ(\alpha\,x-e_n)+\delta y^T\,G_k\,Q(\alpha\,x-e_n) \\ &+y^T\delta G_kQ(\alpha\,x-e_n)+y^TG_k\widetilde{\delta Q}(\alpha\,x-e_n)+y^TG_kQ(\alpha\,\delta x) \end{aligned}$$

for $k = 0,\ldots,n-1$. The largest term on the right-hand side is the second. It contains a factor α^2, and for this reason $|\,\delta a_k\,| \lesssim u\,|\,\alpha\,|^2 = u\,\|a\|^2$, and thus $\|\delta a\| \lesssim u\,\|a\|^2$. ∎

Figure 5.5 verifies that the backward error grows quadratically in $\|a\|$.

The squared norm in Theorem 5.11 is disappointing, but fortunately we can get rid of it by allowing p to be approximated by a non-monic polynomial. Let $\hat{p} = \beta^{-1}\tilde{p} = (1-\alpha\,\delta\rho)\tilde{p}$. This polynomial also has the computed roots as its exact zeros. By (5.11) we have

$$\begin{aligned} \hat{p}(\lambda) &= (1-\alpha\,\delta\rho)\det(\lambda I-(Q+\delta Q)) \\ &\quad +\sum_{k=0}^{n-1}(y+\delta y)^T(G_k+\delta G_k)(Q+\widetilde{\delta Q})(\alpha\,(x+\delta x)-e_n). \end{aligned} \tag{5.12}$$

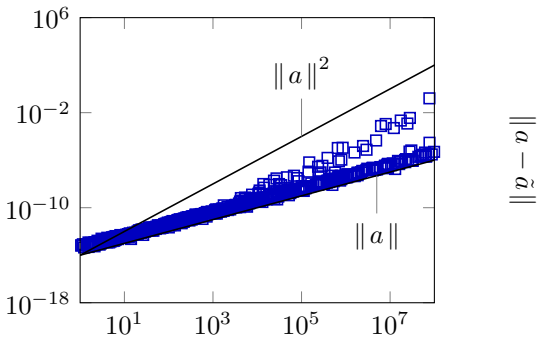


FIG. 5.5. *Backward error of companion QR on the polynomial coefficients, monic* $\tilde{p}$

By expanding this we get the following theorem. Notice that we can get away with the less stringent assumption on $\|a\|$ because we do not need to make a first-order approximation of β.

THEOREM 5.12. *Suppose we apply the companion QR algorithm to the monic polynomial* p *with coefficient vector* a*. Let* $\tilde{p}$ *denote the monic polynomial that has the computed roots as its exact zeros, let* $\hat{p} = (1 - \alpha\,\delta\rho)\tilde{p}$*, and let* $a + \delta a$ *denote the coefficient vector of* $\hat{p}$*. If* $\|a\| \ll 1/u$*, then*

$$\|\delta a\| \lesssim u\,\|a\|.$$

Proof. In the first term of (5.12), let $q_k + \widetilde{\delta q}_k$ denote the kth coefficient of $\det(\lambda I - (Q + \delta Q))$. We have given δq_k the new name $\widetilde{\delta q}_k$. Then we can rewrite the first term in the form

$$(1 - \alpha\,\delta\rho)\sum_{k=0}^{n}(q_k + \widetilde{\delta q}_k)\lambda^k = \sum_{k=0}^{n}(q_k + \alpha\,\delta q_k)\lambda^k,$$

where $\alpha\,\delta q_k \doteq \widetilde{\delta q}_k - \alpha\,\delta\rho\,q_k$ and $|\,\delta q_k\,| \lesssim u$. If we now expand (5.12) we get

$$\begin{aligned}\delta a_k \doteq\ & \alpha\,\delta q_k + \delta y^T\,G_k\,Q(\alpha\,x - e_n)\\ & + y^T\delta G_k Q(\alpha\,x - e_n) + y^T G_k \delta Q(\alpha\,x - e_n) + y^T G_k Q(\alpha\,\delta x)\end{aligned}$$

for $k = 0, \ldots, n$ (where $G_n = 0$). Now there are no terms with a factor α^2; each term has a factor α. We conclude that $\|\delta a\| \lesssim u\,\|a\|$. ∎

Theorem 5.12 shows that in cases where $\|a\|$ is large, and Theorem 5.11 would yield a bad result, most of the error is parallel to p and is thus irrelevant. That part of the error can be removed simply by rescaling $\tilde{p}$.

Example 5.13. As a numerical test of Theorem 5.12 we computed the monic $\tilde{p}$ *and then a projected* $\hat{p} = \gamma\tilde{p}$*, where* γ *is chosen so that the quantity* $\|a - \gamma\tilde{a}\|$ *is minimized. In Figure 5.6 we have plotted* $\|\delta a\| = \|\hat{a} - a\|$ *against* $\|a\|$*. The graph on the left shows the results from our companion QR code, and we observe that the relationship is linear in* $\|a\|$*. The right-hand graph shows the same computation for the unstructured LAPACK code with balancing. We see that in this case the backward error grows like* $\|a\|^2$*. Thus our companion QR algorithm is more accurate than the unstructured algorithm by this measure.*

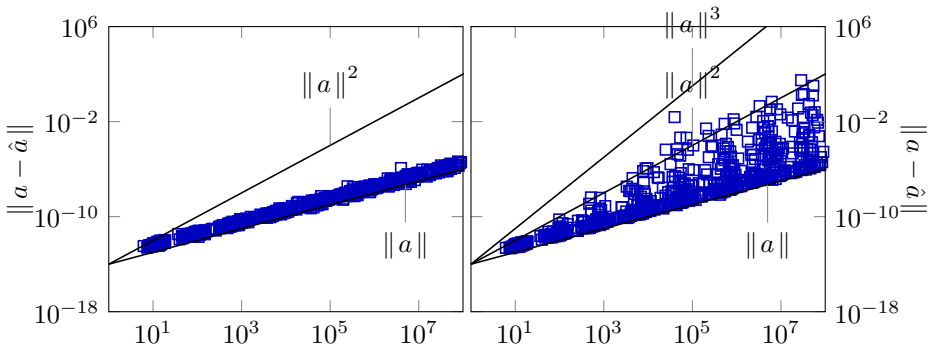


FIG. 5.6. *Backward error on the polynomial coefficients, non-monic $\hat{p}$, companion QR code on left and unstructured LAPACK code on right.*

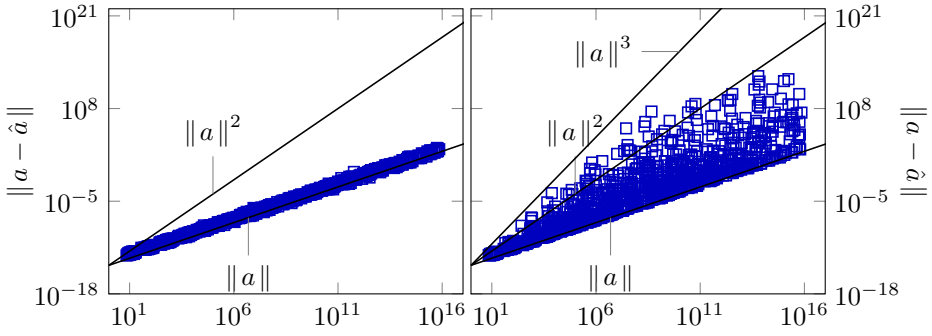


FIG. 5.7. *Backward error on the polynomial coefficients, non-monic $\hat{p}$, in the case of extremely large $\|a\|$.*

All of our theorems have made at least the assumption $\|a\| \ll 1/u$. We have found that in practice the results continue to hold even for very large a that grossly violate this assumption. Figure 5.7 shows the same experiment as Figure 5.6, except that polynomials with norm as large as 10^{20} are allowed.

So far throughout this section we have assumed for convenience that we are dealing with a monic polynomial. In practice we will often have a non-monic p, which we make monic by rescaling it. The following theorem covers this case.

THEOREM 5.14. *Suppose we compute the roots of a non-monic polynomial p with coefficient vector a by applying the companion QR algorithm to the monic polynomial p/a_n with coefficient vector a/a_n. Let $\tilde{p}$ denote the monic polynomial that has the computed roots as its exact zeros, let $\hat{p} = a_n(1 - \alpha\,\delta\rho)\tilde{p}$, and let $a + \delta a$ denote the coefficient vector of $\hat{p}$. If $\|a\|/|a_n| \ll 1/u$, then*

$$\|\delta a\| \lesssim u\,\|a\|.$$

Proof. Apply Theorem 5.12 to p/a_n, then rescale by multiplying by a_n. ∎

5.3. Backward error of the companion QZ algorithm. We now consider the backward error of the companion QZ algorithm, which finds the zeros of a non-

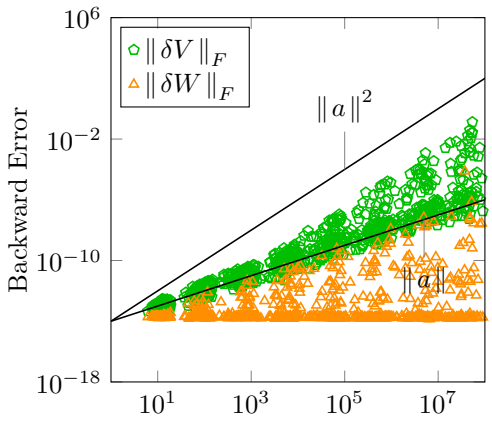


FIG. 5.8. *Backward errors* $\| \delta V \|_F$ *and* $\| \delta W \|_F$ *of companion QZ plotted against* $\| a \|$

monic polynomial

$$p(z) = a_0 + a_1 z + \cdots + a_{n-1} z^{n-1} + a_n z^n$$

by computing the eigenvalues of a pencil $V - \lambda W$ of the form (1.3) with vectors v and w satisfying (1.4). We will assume a reasonable choice of v and w so that $\max\{\| v \|, \| w \|\} \approx \| a \|$, where a is the coefficient vector of p as before. (In fact, in all of our numerical experiments we have made the simplest choice, namely the one given by (1.2).)

Notice that in this setting we have the freedom to rescale the coefficients of the polynomial by an arbitrary factor. Thus we can always arrange to have $\| a \| \approx 1$, for example. This is the advantage of this approach, and this is what allows us to get an optimal backward error bound in this case.

When we run the companion QZ algorithm on (V, W), we obtain

$$\hat{V} = U^*(V + \delta V)Z, \quad \hat{W} = U^*(W + \delta W)Z,$$

where δV and δW are the backward errors. We begin with an analogue of Theorem 5.2.

THEOREM 5.15. *Assume* $\| a \| \lesssim 1/\sqrt{u}$. *If the companion pencil eigenvalue problem is solved by the companion QZ algorithm, then*

(a) $\hat{V} = U^*(V + \delta V)Z$, *where* $\| \delta V \| \lesssim u \| a \|^2$,

(b) $\hat{W} = U^*(W + \delta W)Z$, *where* $\| \delta W \| \lesssim u \| a \|^2$.

Proof. The proof for $V = QR$ is identical to the proof of Theorem 5.2. The proof for W is even simpler because W is already upper triangular; there is no unitary Q factor to take into account. □

Figure 5.8 gives numerical confirmation of Theorem 5.15. We see that the growth is quadratic in $\| a \|$ in the worst case.

It is natural to ask whether we can get a better fit by scaling the pencil as we did in the proof of Theorem 5.5. The answer seems to be no.

Example 5.16. We computed the backward errors and then computed γ *to minimize* $(\| \gamma(V + \delta V) - V \|_F^2 + \| \gamma(W + \delta W) - W \|_F^2)^{1/2}$, *which is a measure of distance between* $\gamma(V + \delta V, W + \delta W)$ *and* (V, W). *The result is shown in Figure 5.9. We observe that this measure of error also grows quadratically with* $\| a \|$.

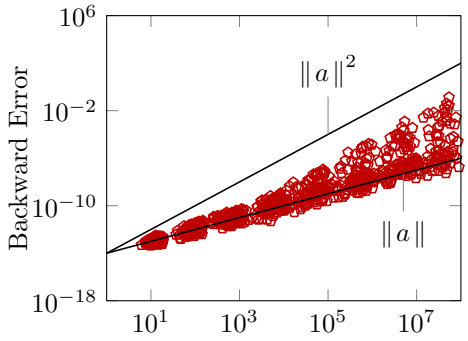


FIG. 5.9. *Backward error* $(\|\gamma(V+\delta V)-V\|_F^2+\|\gamma(W+\delta W)-W\|_F^2)^{1/2}$ *(optimal* γ*) plotted against* $\|a\|$

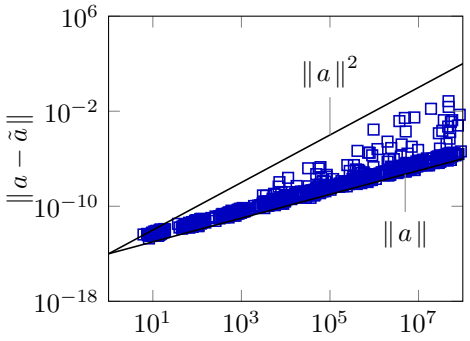


FIG. 5.10. *Backward error* $\|a-\hat{a}\|$ *for companion QZ*

Now let us consider the backward error of the companion QZ algorithm on the polynomial coefficient vector a. We will compute an optimally scaled backward error as in Example 5.13: Given the computed roots, we build the monic polynomial $\tilde{p}$ (with coefficient vector $\tilde{a}$) that has these as its exact roots. We then let $\hat{p}=\gamma\tilde{p}$ (with coefficient vector $\hat{a}$), where γ is chosen so that $\|a-\gamma\,\tilde{a}\|$ is minimized. We hope to get a backward error $\|a-\hat{a}\|$ that is linear in $\|a\|$, as in Theorem 5.12, but Figure 5.10 shows what we actually get. The backward error seems to grow quadratically in $\|a\|$, which is a disappointment.

THEOREM 5.17. *Under the assumption* $\|a\|\lesssim 1/\sqrt{u}$*, the backward error of the companion QZ algorithm on the polynomial coefficient vector satisfies*

$$\|a-\hat{a}\|\lesssim u\,\|a\|^2.$$

Proof. This is similar to the proof of Theorem 5.11. There we made use of the decomposition $A=QR$, where $R=I+(\alpha\,x-e_n)y^T$. We can do the same here with the matrix V. $V=QR$, where $R=I+(\alpha\,x-e_n)y^T$, $y=e_n$, $|\alpha|=\|\underline{v}\|\lesssim\|a\|$, and $\|x\|\leq 1$. W has a similar form: $W=I+(\check{\alpha}\,\check{x}-e_n)y^T$, where $y=e_n$, $|\check{\alpha}|=\|\underline{w}\|\lesssim\|a\|$, and $\|\check{x}\|\leq 1$. Thus

$$\lambda W-V=(\lambda I-Q)+(\lambda(\check{\alpha}\,\check{x}-e_n)-Q(\alpha\,x-e_n))y^T. \tag{5.13}$$

Applying Lemma 5.7 we find that the characteristic polynomial $p(\lambda) = \det(\lambda W - V)$ is given by

$$p(\lambda) = \det(\lambda I - Q) + y^T \operatorname{adj}(\lambda I - Q)(\lambda(\check{\alpha}\,\check{x} - e_n) - Q(\alpha\, x - e_n)).$$

If we now note that $\det(\lambda I - Q) = \lambda^n - 1$ and substitute for $\operatorname{adj}(\lambda I - Q)$ using (5.4), we obtain

$$p(\lambda) = \lambda^n - 1 + \sum_{k=0}^{n} y^T (G_{k-1}(\check{\alpha}\,\check{x} - e_n) - G_k Q(\alpha\, x - e_n))\lambda^k,$$

where $G_{-1} = G_n = 0$. Thus the kth coefficient of p is

$$a_k = q_k + y^T (G_{k-1}(\check{\alpha}\,\check{x} - e_n) - G_k Q(\alpha\, x - e_n)), \tag{5.14}$$

where $q_n = 1$, $q_0 = -1$, and $q_k = 0$ otherwise.

Now let's look at the backward error. Lemma 5.9 holds for the matrix R in the decomposition $V = QR$. So, just as in the beginning lines of the proof of Theorem 5.11, we have

$$V + \delta V = (Q + \delta Q) + \beta\,(Q + \widetilde{\delta Q})(\alpha(x + \delta x) - e_n)(y + \delta y)^T,$$

where $\|\,\delta Q\,\| \lesssim u$, $\|\,\widetilde{\delta Q}\,\| \lesssim u$, $\|\,\delta x\,\| \lesssim u$, and $\|\,\delta y\,\| \lesssim u$. Lemma 5.9 also holds for the matrix W, so

$$W + \delta W = (I + \check{\delta} I) + \check{\beta}(\check{\alpha}(\check{x} + \check{\delta} x) - e_n)(y + \check{\delta} y)^T,$$

where $\|\,\check{\delta} I\,\| \lesssim u$ $\check{\beta} = (1 - \check{\alpha}\check{\delta}\rho)^{-1}$, $|\,\check{\delta}\rho\,| \lesssim u$, $\|\,\check{\delta} x\,\| \lesssim u$, and $\|\,\check{\delta} y\,\| \lesssim u$. Putting this all together we get the perturbed pencil

$$\begin{aligned}\lambda(W + \delta W) - (V + \delta V) = {} & \lambda(I + \check{\delta} I) - (Q + \delta Q) \\ & + \lambda\,\check{\beta}\,(\check{\alpha}(\check{x} + \check{\delta} x) - e_n)(y + \check{\delta} y)^T \\ & - \beta\,(Q + \widetilde{\delta Q})(\alpha(x + \delta x) - e_n)(y + \delta y)^T.\end{aligned}$$

The low-rank part of this pencil has rank two because $y + \check{\delta} y \neq y + \delta y$. We would like it to have rank one because we want to apply Lemma 5.7, for which we need a rank-one perturbation. Notice that we can make it have rank one by perturbing it slightly. Specifically we can subtract off the small quantity

$$\lambda\, S = \lambda\,\check{\beta}\,(\check{\alpha}(\check{x} + \check{\delta} x) - e_n)(\check{\delta} y - \delta y)^T.$$

The assumption $\|\,a\,\| \lesssim 1/\sqrt{u}$ implies that $|\,\check{\alpha}\,\check{\delta}\rho\,| \ll 1$, so $\check{\beta} = (1 - \check{\alpha}\,\check{\delta}\rho)^{-1} \approx 1$. Also $|\,\check{\alpha}\,| \lesssim \|\,a\,\|$ and $\|\,\check{\delta} y - \delta y\,\| \lesssim u$, so

$$\|\,S\,\| \lesssim u\,\|\,a\,\|.$$

The subtraction of $\lambda\, S$ from the low-rank part will be offset by adding $\lambda\, S$ to the term $\lambda\,\check{\delta} I$. Define

$$\delta I = \check{\delta} I + S.$$

Then $\|\,\delta I\,\| \lesssim u\,\|\,a\,\|$ and, defining the auxiliary quantity

$$v = \lambda\,\check{\beta}\,(\check{\alpha}(\check{x} + \check{\delta} x) - e_n) - \beta\,(Q + \widetilde{\delta Q})(\alpha(x + \delta x) - e_n), \tag{5.15}$$

we have

$$\lambda(W+\delta W)-(V+\delta V)=\lambda(I+\delta I)-(Q+\delta Q)+v(y+\delta y)^T.$$

The roots computed by companion QZ are the exact roots of

$$\tilde{p}(\lambda)=\det(\lambda(W+\delta W)-(V+\delta V)).$$

Replacing (V,W) by $(V+\delta V, W+\delta W)$ in (5.13), taking the determinant, and using Lemma 5.7, we get

$$\tilde{p}(\lambda)=\det(\lambda(I+\delta I)-(Q+\delta Q))+(y+\delta y)^T\operatorname{adj}(\lambda\,(I+\delta I)-(Q+\delta Q))v. \quad (5.16)$$

We deal with the first term by appealing to [14, 24], just as in the proof of Theorem 5.11. We deduce that $\det(\lambda(I+\delta I)-(Q+\delta Q))=\sum_{k=0}^{n}(q_k+\delta q_k)\lambda^k$, where $|\,\delta q_k\,|\lesssim u\,\|\,a\,\|$. The factor $\|\,a\,\|$ shows up here because $\|\,\delta I\,\|\lesssim u\,\|\,a\,\|$.

For the term with the adjugate we invoke Lemma 5.10 with $\epsilon=u\,\|\,a\,\|$ to deduce that

$$\operatorname{adj}(\lambda\,(I+\delta I)-(Q+\delta Q))=\sum_{k=0}^{n-1}(G_k+\delta G_k)\lambda^k$$

with $\|\,\delta G_k\,\|\lesssim u\,\|\,a\,\|$.

Making these substitutions into (5.16) and using the expression (5.15) for v we get

$$\tilde{p}(\lambda)=\sum_{k=0}^{n}\tilde{a}_k\,\lambda^k,$$

where

$$\begin{aligned}\tilde{a}_k &= q_k+\delta q_k\\ &\quad+(y+\delta y)^T(G_{k-1}+\delta G_{k-1})\,\check{\beta}\,(\check{\alpha}\,(\check{x}+\check{\delta x})-e_n)\\ &\quad-(y+\delta y)^T(G_k+\delta G_k)(Q+\widetilde{\delta Q})\,\beta\,(\alpha\,(x+\delta x)-e_n).\end{aligned} \quad (5.17)$$

Again we have $G_{-1}=G_n=0$. This equation should be compared to the equation (5.14) for the coefficients of the unperturbed polynomial. Notice the extra factors β and $\check{\beta}$. The assumption $\|\,a\,\|\lesssim 1/\sqrt{u}$ allows us to make the approximations $\beta\doteq 1+\alpha\,\delta\rho$ and $\check{\beta}\doteq 1+\check{\alpha}\,\check{\delta\rho}$. If we do this and expand (5.17), dropping second-order terms, we get

$$\begin{aligned}\tilde{a}_k-a_k \doteq\ & \delta q_k+\delta y^T(G_{k-1}(\check{\alpha}\,\check{x}-e_n)-G_kQ(\alpha\,x-e_n))\\ &+y^T(\delta G_{k-1}(\check{\alpha}\,\check{x}-e_n)-\delta G_kQ(\alpha\,x-e_n)) \quad (5.18)\\ &-y^TG_k\,\widetilde{\delta Q}(\alpha\,x-e_n)\\ &+y^T(G_{k-1}\check{\alpha}\,\check{\delta\rho}(\check{\alpha}\,\check{x}-e_n)-G_kQ\,\alpha\,\delta\rho(\alpha\,x-e_n)) \quad (5.19)\\ &+y^T(G_{k-1}\check{\alpha}\,\check{\delta x}-G_kQ\,\alpha\,\delta x).\end{aligned}$$

The terms in line (5.18) are of order $u\,\|\,a\,\|^2$ because $\|\,\delta G_k\,\|\lesssim u\,\|\,a\,\|$, $|\,\alpha\,|\lesssim\|\,a\,\|$, and $|\,\check{\alpha}\,|\lesssim\|\,a\,\|$. The terms in line (5.19) are also of order $\|\,a\,\|^2$ because of the factors α^2

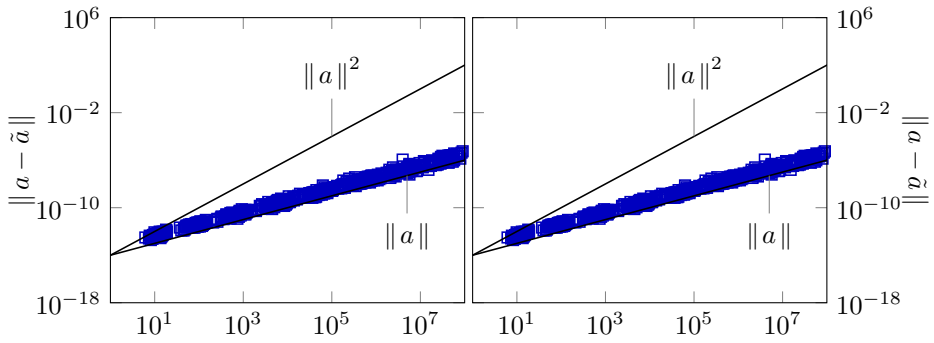


FIG. 5.11. *Backward error of scaled companion QZ algorithm (left) and unstructured QZ algorithm (right)*

and $\check{\alpha}^2$. All of the other terms are of order $u\,\|a\|$. We deduce that $|\tilde{a}_k - a_k| \lesssim u\,\|a\|^2$, $k = 0, 1, \ldots, n$. Thus $\|\tilde{a} - a\| \lesssim u\,\|a\|^2$. ∎

In the companion QR case we were able to bring the backward error down to $u\,\|a\|$ by rescaling $\tilde{p}$. Figure 5.10 suggests that this will not be possible in the companion QZ case. The reader might wonder why not. In the companion QR case we got a better fit by dividing $\tilde{p}$ by β. In the companion QZ case we have a β and a $\check{\beta}$, and they are different. We can't divide by both of them at once.

So far it looks like companion QR is (surprisingly) more accurate than companion QZ, but we have not yet taken into account the freedom to rescale that we have in the companion QZ case.

THEOREM 5.18. *Suppose we compute the zeros of the polynomial p with coefficient vector a by applying the companion QZ algorithm to polynomial $p/\|a\|$ with coefficient vector $a/\|a\|$. Then the backward error satisfies*

$$\|a - \hat{a}\| \lesssim u\,\|a\|.$$

Proof. By Theorem 5.17 the backward error on $b = a/\|a\|$ satisfies $\|b - \hat{b}\| \lesssim u\,\|b\|^2 = u$. Therefore the backward error on a, which is $a - \hat{a} = \|a\|(b - \hat{b})$ satisfies $\|a - \hat{a}\| = \|a\|\,\|b - \hat{b}\| \lesssim u\,\|a\|$. ∎

In the interest of full disclosure we must point out that this argument applies equally well to any stable method for computing the eigenvalues of the pencil. If we use, for example, the unstructured QZ algorithm on the rescaled polynomial $p/\|a\|$, we will get the same result. Figure 5.11 provides numerical confirmation of Theorem 5.18. In the left panel we the backward error for companion QZ, and in the right panel we have the backward error of the unstructured QZ code from LAPACK.

6. Conclusions. The companion QR algorithm is more accurate than the unstructured QR algorithm even if a turnover that does not guarantee high relative accuracy of ρ is used. Simpler and stronger results can be obtained if the turnover preserves high relative accuracy of ρ. These are presented in [3].

REFERENCES

[1] J. Aurentz, R. Vandebril, and D. S. Watkins, *Fast computation of the zeros of a polynomial via factorization of the companion matrix*, SIAM Journal on Scientific Computing, 35 (2013), pp. A255–A269.
[2] ———, *Fast computation of eigenvalues of companion, comrade, and related matrices*, BIT Numerical Mathematics, 54 (2014), pp. 7–30.
[3] J. L. Aurentz, T. Mach, L. Robol, R. Vandebril, and D. S. Watkins, *Fast and backward stable computation of roots of polynomials, part ii: backward error analysis; companion matrix and companion pencil*. submitted for publication, 2017.
[4] J. L. Aurentz, T. Mach, R. Vandebril, and D. S. Watkins, *Fast and backward stable computation of roots of polynomials*, SIAM Journal on Matrix Analysis and Applications, 36 (2015), pp. 942–973.
[5] ———, *Fast and stable unitary QR algorithm*, Electronic Transactions on Numerical Analysis, 44 (2015), pp. 327–341.
[6] J. L. Aurentz, T. Mach, R. Vandebril, and D. S. Watkins, *A note on companion pencils*, Contemporary Mathematics, 658 (2016), pp. 91–101.
[7] D. A. Bini, P. Boito, Y. Eidelman, L. Gemignani, and I. Gohberg, *A fast implicit QR eigenvalue algorithm for companion matrices*, Linear Algebra and its Applications, 432 (2010), pp. 2006–2031.
[8] P. Boito, Y. Eidelman, and L. Gemignani, *Implicit QR for companion-like pencils*, Mathematics of Computation, 85 (2016), pp. 1753–1774.
[9] P. Boito, Y. Eidelman, and L. Gemignani, *A real QZ algorithm for structured companion pencils*. https://arxiv.org/abs/1608.05395, 2016.
[10] P. Boito, Y. Eidelman, L. Gemignani, and I. Gohberg, *Implicit QR with compression*, Indagationes Mathematicae, 23 (2012), pp. 733–761.
[11] S. Chandrasekaran, M. Gu, J. Xia, and J. Zhu, *A fast QR algorithm for companion matrices*, Operator Theory: Advances and Applications, 179 (2007), pp. 111–143.
[12] F. De Terán, F. M. Dopico, and J. Pérez, *Backward stability of polynomial root-finding using Fiedler companion matrices*, IMA Journal of Numerical Analysis, 36 (2016), pp. 133–173.
[13] B. Eastman, I.-J. Kim, B. Shader, and K. Vander Meulen, *Companion matrix patterns*, Linear Algebra and its Applications, 463 (2014), pp. 255–272.
[14] A. Edelman and H. Murakami, *Polynomial roots from companion matrix eigenvalues*, Mathematics of Computation, 64 (1995), pp. 763–776.
[15] J. G. F. Francis, *The QR transformation, part II*, Computer Journal, 4 (1961), pp. 332–345.
[16] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, Second ed., 2013.
[17] G. F. Jónsson and S. Vavasis, *Solving polynomials with small leading coefficients*, SIAM Journal on Matrix Analysis and Applications, 26 (2004), pp. 400–414.
[18] D. S. Mackey, N. Mackey, C. Mehl, and V. Mehrmann, *Vector spaces of linearizations for matrix polynomials*, SIAM Journal on Matrix Analysis and Applications, 28 (2006), pp. 971–1004.
[19] J. M. McNamee, *A bibliography on roots of polynomials*, Journal of Computational and Applied Mathematics, 47 (1993), pp. 391–394.
[20] C. B. Moler and G. W. Stewart, *An algorithm for generalized matrix eigenvalue problems*, SIAM Journal on Numerical Analysis, 10 (1973), pp. 241–256.
[21] *MPFUN - multiprecision software*. http://www.netlib.org/mpfun/, 2005.
[22] B. N. Parlett and C. Reinsch, *Balancing a matrix for calculation of eigenvalues and eigenvectors*, Numerische Mathematik, 13 (1969), pp. 293–304.
[23] G. W. Stewart, *On the adjugate matrix*, Linear Algebra Appl., 283 (1998), pp. 151–164.
[24] P. Van Dooren and P. Dewilde, *The eigenstructure of an arbitrary polynomial matrix: Computational aspects*, Linear Algebra and its Applications, 50 (1983), pp. 545–579.
[25] R. Vandebril and D. S. Watkins, *An extension of the QZ algorithm beyond the Hessenberg-upper triangular pencil*, Electronic Transactions on Numerical Analysis, 40 (2012), pp. 17–35.
[26] D. S. Watkins, *The Matrix Eigenvalue Problem: GR and Krylov Subspace Methods*, SIAM, Philadelphia, USA, 2007.
[27] D. S. Watkins, *Fundamentals of Matrix Computations*, John Wiley & Sons, Inc., New York, Third ed., 2010.